User’s Guide

3HAC 0966-21
For BaseWare OS 3.1

ABB Flexible Automation

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Robotics Products AB. ABB Robotics Products AB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Robotics Products AB be liable for incidental or consequential damages arising from use of this document or of the software and hardware described in this document.

This document and parts thereof must not be reproduced or copied without
ABB Robotics Products AB´s written permission, and contents thereof must not be imparted to a third party nor be used for any unauthorized purpose. Contravention will be prosecuted.

Additional copies of this document may be obtained from ABB Robotics Products AB at its then current charge.

© ABB Robotics Products AB

Article number: 3HAC 0966-21
Issue: For BaseWare OS 3.1

ABB Robotics Products AB
S-721 68 Västerås
Sweden

CONTENTS

Page

1 Table of Contents 1-1

2 Introduction 2-1

1 New Features in this Version of the Robot 2-3

2 Other Manuals 2-3

3 How to Read this Manual 2-3

3.1 Typographic conventions 2-4

4 Reader's Comments 2-5

3 Safety 3-1

1 General 3-3

1.1 Introduction 3-3

2 Applicable Safety Standards 3-3

3 Fire-Extinguishing 3-4

4 Definitions of Safety Functions 3-4

5 Safe Working Procedures 3-5

5.1 Normal operations 3-5

6 Programming, Testing and Servicing 3-5

7 Safety Functions 3-6

7.1 The safety control chain of operation 3-6

7.2 Emergency stops 3-7

7.3 Mode selection using the operating mode selector 3-7

7.4 Enabling device 3-8

7.5 Hold-to-run control 3-8

7.6 General Mode Safeguarded Stop (GS) connection 3-9

7.7 Automatic Mode Safeguarded Stop (AS) connection 3-10

7.8 Limiting the working space 3-10

7.9 Supplementary functions 3-10

8 Safety Risks Related to End Effectors 3-10

8.1 Gripper 3-10

8.2 Tools/workpieces 3-11

8.3 Pneumatic/hydraulic systems 3-11

9 Risks during Operation Disturbances 3-11

10 Risks during Installation and Service 3-11

11 Risks Associated with Live Electric Parts 3-12

12 Emergency Release of Mechanical Arm 3-13

13 Limitation of Liability 3-13

14 Related Information 3-13

Page

4 Basic Operation 4-1

5 Starting up 5-1

1 Switching on the Power Supply 5-3

1.1 Errors on start-up 5-4

2 The Operator's Panel 5-4

3 Selecting the Operating Mode 5-4

3.1 Automatic mode (production mode) 5-4

3.2 Manual mode with reduced speed (programming mode) 5-5

3.3 Manual mode with full speed (testing mode) 5-5

4 Switching the Power Supply to the Motors On 5-5

5 Emergency Stops 5-6

5.1 Activating the emergency stop button 5-6

5.2 Resetting after an emergency stop 5-6

6 The Teach Pendant 5-7

6.1 Entering text using the teach pendant 5-9

6 Jogging 6-1

1 General 6-3

1.1 The Jogging window 6-3

1.2 Reading the current position 6-4

1.3 How moving the joystick affects movements 6-4

1.4 Locking of joystick axes 6-5

1.5 Motion Supervision 6-5

2 Jogging the Robot 6-6

2.1 Jogging the robot along one of the base coordinate axes 6-6

2.2 Jogging the robot in the direction of the tool 6-7

2.3 Reorienting the tool 6-9

2.4 Aligning a tool along a coordinate axis 6-9

2.5 Jogging the robot in the direction of the work object 6-11

2.6 Jogging the robot along one of the world coordinate axes 6-13

2.7 Using a stationary tool 6-13

2.8 Jogging the robot axis-by-axis 6-14

2.9 Incremental movement 6-14

2.10 Jogging an unsynchronised axis 6-15

3 Jogging External Axes 6-16

3.1 Choosing external units 6-16

3.2 Jogging external units axis-by-axis 6-16

Page

3.3 Jogging external units coordinated 6-17

7 Inputs and Outputs 7-1

1 General 7-3

1.1 The Inputs/Outputs window 7-3

1.2 Choosing an I/O list 7-4

1.3 Defining the *Most Common* I/O list 7-4

2 Changing Signal Values 7-6

2.1 Changing the value of a digital output 7-6

2.2 Changing the value of an analog output signal or a group of output signals 7-6

3 Displaying Information 7-7

3.1 To display information on a given signal 7-7

3.2 To display a chart of all digital signals of a unit 7-7

3.3 To print an I/O list 7-8

8 Programming and Testing 8-1

1 Creating a New Program 8-5

1.1 What is a program? 8-5

1.2 The Program window 8-6

1.3 Creating a new program 8-6

1.4 Loading an existing program 8-7

2 Defining Tools and Work Object 8-7

3 Creating New Routines 8-8

3.1 What is a routine? 8-8

3.2 The *Program Routines* window 8-9

3.3 Creating a new routine 8-10

3.4 Duplicating a routine 8-11

4 Creating new instructions 8-12

4.1 Choosing a routine 8-12

4.2 The *Program Instr* window 8-12

4.3 What is an instruction? 8-13

4.4 Getting more information about an instruction 8-14

5 Programming 8-14

5.1 Choosing from the instruction pick list 8-15

5.2 Adding an instruction 8-16

5.3 Expressions 8-18

5.4 Moving and copying instructions 8-21

6 Running Programs 8-21

Page

6.1 Program execution 8-21
6.2 The *Program Test* window 8-22
6.3 Choosing the speed correction 8-22
6.4 Choosing the execution mode 8-23
6.5 Starting program execution 8-24
6.6 Stopping program execution 8-25
6.7 Where will the program start? 8-25
6.8 Simulating wait conditions 8-27
7 Saving and Printing Programs 8-28
7.1 Saving the program on diskette or some other type of mass memory 8-28
7.2 Printing a program from the robot 8-29
7.3 Printing a program using a PC 8-29
8 Changing the Program 8-29
8.1 Selecting an instruction or an argument 8-30
8.2 Modifying the position in a positioning instruction 8-31
8.3 Tuning position during program execution 8-31
8.4 Changing an argument 8-33
8.5 Adding optional arguments 8-34
8.6 Changing the structure of an IF, FOR or TEST instruction 8-35
8.7 Changing the name or declaration of a routine 8-35
8.8 Deleting an instruction or an argument 8-36
8.9 Deleting a routine 8-36
8.10 Undo latest action 8-36
9 Special Editing Functions 8-37
9.1 Search & replace 8-37
9.2 Mirroring 8-39
10 Creating Data 8-45
10.1 What is data? 8-45
10.2 The *Program Data* window (used to manage data) 8-45
10.3 Creating new data 8-47
10.4 Creating new array data 8-48
10.5 Duplicating data 8-50
10.6 Storing position data using the robot 8-50
10.7 Routine data 8-50
11 Changing Data 8-50
11.1 Viewing and possibly changing the current value 8-50
11.2 Changing data names or declarations 8-51

Page

11.3 Deleting data ... 8-52
12 Error Handling ... 8-52
13 Using Modules ... 8-54
13.1 What is a module? ... 8-54
13.2 Choosing modules ... 8-55
13.3 Creating a new module ... 8-56
13.4 Changing the name or declaration of a module ... 8-56
13.5 Reading a program module from diskette or some other type of mass memory ... 8-57
13.6 Deleting program modules from the program ... 8-57
13.7 Listing all routines in all modules ... 8-57
13.8 Duplicating a routine from one module to another ... 8-58
13.9 Listing all data in the current module ... 8-58
13.10 Duplicating data from one module to another ... 8-58
13.11 Saving modules on diskette or some other type of mass memory ... 8-58
13.12 Calling up the complete module list ... 8-59
14 Preferences ... 8-60
14.1 Defining the *Most Common* instruction pick list ... 8-60
14.2 Default data Global/Local ... 8-61
14.3 Defining programming rule for robot positions ... 8-62
9 The programming language RAPID ... 9-1
1 Programming a Position ... 9-3
1.1 Positioning instructions ... 9-3
1.2 Programming an offset ... 9-6
2 Changing the Value of an Output ... 9-7
3 Waiting ... 9-8
3.1 Wating for an input ... 9-8
3.2 Waiting a specific amount of time ... 9-10
4 Controlling the Program Flow ... 9-10
4.1 Calling a subroutine ... 9-10
4.2 Program control within a routine ... 9-11
5 Assigning a Value to Data (Registers) ... 9-14
10 Calibration ... 10-1
1 Coordinate systems ... 10-3
2 Coordinated axes ... 10-5
2.1 External axes, general ... 10-5
2.2 Coordination ... 10-5

Page

3 Calibration 10-6
3.1 What is calibration? 10-6
3.2 Viewing the calibration status 10-6
3.3 Checking the calibration 10-7
3.4 Updating revolution counters 10-8
4 Base Frame for the Robot 10-9
4.1 Defining the Base Frame for the Robot 10-9
5 Coordinated track motion 10-12
5.1 How to get started with a coordinated track motion 10-12
5.2 Defining the Base Frame for a track motion 10-12
6 Coordinated external axes 10-16
6.1 How to get started with a coordinated (moveable) user coordinate system . 10-16
6.2 Defining the User Frame for a rotational axis (single) 10-17
6.3 Defining the User Frame for a two-axes mechanical unit, Method 1 10-20
6.4 Defining the User Frame for a two-axes mechanical unit, Method 2 10-23
7 Defining Tools 10-28
7.1 Creating a new tool 10-28
7.2 Manually updating the TCP and weight of a tool 10-29
7.3 Methods of defining the tool coordinate system 10-29
7.4 Using the robot to change the TCP and orientation of a tool 10-31
7.5 Stationary tool 10-33
8 Work Objects and Program Displacements 10-35
8.1 General 10-35
8.2 Using work objects 10-36
8.3 Creating a new work object 10-36
8.4 Manually updating the user and object coordinate system of the work object 10-37
8.5 Methods of defining a work object 10-37
8.6 Using the robot to change the work object 10-38
8.7 Defining a moveable object frame 10-40
8.8 How to use different work objects to get different displacements 10-40
8.9 How to adjust the program vertically using the object frame 10-42
8.10 Using program displacement 10-42
8.11 Creating a new displacement frame 10-43
8.12 Manually updating a displacement frame 10-43
8.13 Methods for defining a displacement frame 10-44
8.14 Using the robot to change a displacement frame 10-44

Page

11 Production Running 11-1
1 The Production Window 11-3
2 Reading a Program 11-4
3 Changing the Override Speed 11-5
4 Changing the Program Running Mode 11-5
5 Starting the Program 11-6
5.1 Restarting after a stop 11-7
5.2 Starting a program from the beginning 11-7
6 Stopping the Program 11-7
7 Tuning position 11-8
8 Operator Dialogs 11-9
12 System Parameters 12-1
1 Changing a Parameter 12-3
1.1 Subdivision of parameters 12-3
1.2 Changing a parameter 12-3
1.3 Deleting a parameter 12-4
1.4 Generating a restart 12-4
1.5 Viewing the last changes that were made 12-5
1.6 Checking Parameters 12-5
2 Saving and Loading Parameters 12-6
2.1 Saving parameters to diskette or some other mass storage device 12-6
2.2 Loading parameters from a diskette or some other mass storage device 12-7
3 Topic: IO Signals 12-9
3.1 Defining I/O Units 12-9
3.2 Additional parameters for gateway (field bus) units 12-10
3.3 Defining input and output signals 12-12
3.4 Defining signal groups 12-14
3.5 Defining cross connections 12-15
3.6 List all available I/O Unit Types 12-19
3.7 I/O Data Specifications 12-21
3.8 Defining system inputs 12-22
3.9 Defining system outputs 12-24
3.10 PLC Communication 12-26
4 Topic: Communication 12-31
4.1 Defining physical channels 12-31
4.2 Defining Transmission Protocol 12-32

Page

4.3 Defining Application Protocol 12-34
5 Topic: Controller 12-37
5.1 Activate delayed safeguarded space stop 12-37
5.2 Activate Hold-To-Run Control 12-37
5.3 Defining event routines 12-38
5.4 Specifying regain distances 12-39
5.5 System miscellaneous 12-40
5.6 Automatic loading of modules and programs 12-41
5.7 Defining multitasking 12-42
6 Topic: TeachPendant 12-45
6.1 Defining Optional Packages 12-45
6.2 Defining File Extension 12-45
6.3 Defining authorisation and confirmation 12-46
6.4 Activation of Limited ModPos Function 12-49
6.5 Programmable keys 12-50
6.6 Defining Running Mode Settings 12-51
7 Topic: Manipulator 12-53
7.1 Defining the commutation offset and calibration offset of the motors 12-53
7.2 Defining the range of movement and calibration position of each axis 12-54
7.3 Defining supervision level 12-54
7.4 Tuning the motion supervision 12-55
7.5 Defining teach mode speed 12-56
7.6 Defining independent motion 12-56
7.7 Defining arm load 12-57
7.8 Defining arm check point 12-58
7.9 Defining external torque 12-59
7.10 Defining friction compensation 12-60
7.11 Defining the base coordinate system 12-61
7.12 Defining external manipulators with more than one axis 12-62
7.13 Defining a track motion with coordinated motion 12-63
7.14 Defining an external mechanical unit coordinated with the robot 12-63
7.15 Defining external axes 12-63
7.16 Activate forced gain control for an external axis 12-75
7.17 Activate notch filter for an external axis 12-76
7.18 Soft servo for external axis 12-77
7.19 Defining the joystick directions for the robot and external manipulator 12-78

7.20 Defining the joystick directions for a single external axis 12-80
7.21 Defining kinematic parameters for general kinematics 12-81
7.22 Servo parameters 12-83
7.23 CPU Optimization 12-84
7.24 Installation optimization of drive system parameters 12-86
13 File Manager 13-1
1 Program/Data Storage 13-3
2 The FileManager Window 13-4
2.1 Choosing a directory 13-4
2.2 Viewing file information 13-4
3 Creating or Moving Files and Directories 13-5
3.1 Creating a new directory 13-5
3.2 Renaming a file or a directory 13-5
3.3 Deleting a file or directory 13-6
3.4 Copying files and directories 13-6
3.5 Moving files and Directories 13-7
3.6 Printing files 13-7
4 Formatting a Diskette 13-7
14 Service 14-1
1 The Service Window 14-3
2 Changing the Current Date and Time 14-3
3 Logs 14-4
3.1 What is a log? 14-4
3.2 What types of logs are there? 14-4
3.3 Viewing all logs 14-5
3.4 Viewing a message in a log 14-6
3.5 Erasing the contents of a log 14-6
3.6 Erasing the contents of all logs 14-6
3.7 Updating the contents of a log automatically or by means of a command 14-7
3.8 Avoiding normal error reports 14-7
3.9 Saving log messages on diskette or some other mass storage device 14-7
4 Calibration 14-8
4.1 What is calibration? 14-8
5 Commutation 14-9
5.1 What is commutation? 14-9
6 Frame Definition 14-9

Page

7 Two Axes Definition 14-9
8 Obtaining information on the robot system 14-9
9 Backup and Restore 14-10
9.1 Perform a Backup 14-10
9.2 Perform a Restore 14-11
10 Perform a Restart 14-11
15 ScreenViewer 15-1
11 User screen 15-3
12 The ScreenViewer Window 15-3
13 The Screen Options 15-4
14 The Screen Loading 15-4
15 The Screen Information 15-4
16 The Screen Display 15-4
16 Error Management 16-1
1 Error Management 16-3
1.1 Confirming an error message 16-3
1.2 Calling up suggestions on how to correct an error 16-3
1.3 Acknowledging warning messages 16-4
16 System and Error Messages 16-5
1 Operational error messages 16-7
2 System error messages 16-9
3 Hardware error messages 16-16
4 Program error messages 16-31
5 Motion error messages 16-58
6 Operator error messages 16-68
7 IO & Communication error messages 16-72
8 Arcweld error messages 16-83
9 Spotweld error messages 16-92
10 Paint error messages 16-93
17 Program Examples 17-1
1 Simple Material Handling 17-1
1.1 What the robot does 17-3
1.2 The main routine 17-3
1.3 Operating the gripper 17-3
1.4 Fetching a part from the In feeder 17-4
1.5 Leaving the part in the machine 17-4

Page

1.6 Starting to process 17-5
1.7 Fetching the part from the machine 17-5
1.8 Leaving the part on the Out feeder 17-5
2 Material Handling 17-7
2.1 What the robot does 17-7
2.2 The main routine 17-7
2.3 Operating the gripper 17-8
2.4 Starting production 17-9
2.5 Fetching the part from the In feeder 17-9
2.6 Leaving the part in the machine 17-9
2.7 Updating operating statistics 17-10
2.8 Stopping production for the day 17-10
18 Quick Reference 18-1
1 The Jogging Window 18-3
1.1 Window: Jogging 18-3
2 The Inputs/Outputs Window 18-4
2.1 Window: Inputs/Outputs 18-4
3 The Program Window 18-6
3.1 Moving between different parts of the program 18-6
3.2 General menus 18-7
3.3 Window: Program Instr 18-10
3.4 Window: Program Routines 18-11
3.5 Window: Program Data 18-13
3.6 Window: Program Data Types 18-15
3.7 Window: Program Test 18-16
3.8 Window: Program Modules 18-18
4 The Production Window 18-19
4.1 Window: Production 18-19
5 The FileManager 18-21
5.1 Window: FileManager 18-21
6 The Service Window 18-23
6.1 General menus 18-23
6.2 Window Service Log 18-26
6.3 Window Service Calibration 18-27
6.4 Window Service Commutation 18-28
7 The System Parameters 18-29
7.1 Window: System Parameters 18-29

Page

19 Special Functionality in this Robot 19-1
20 Index, Glossary 20-1

CONTENTS

Page

1 New Features in this Version of the Robot 3
2 Other Manuals 3
3 How to Read this Manual 3
3.1 Typographic conventions 4
4 Reader's Comments 5

Introduction

This manual will help you whenever you use the robot. It provides step-by-step instructions on how to perform various tasks, such as how to move the robot manually, how to program, or how to start a program when running production.

1 New Features in this Version of the Robot

New functionality and other interesting information can be read from the file *readme* on the Set-up diskette. This file is continuously updated with the latest information, which is why two robots of the same version may contain different information in their *readme* files.

This file can be read from a normal PC, using any word processing program. It can also be loaded into the robot program memory and then read on the teach pendant. For information on how to load programs from diskette, see *Chapter 8: Programming and Testing*.

2 Other Manuals

Before using the robot for the first time, you should read *Basic Operation*. This will provide you with the basics of operating and programming the robot.
Basic Operation is included in this manual, see *Chapter 4*.

The *Product Manual* describes how to install the robot, as well as maintenance procedures and troubleshooting. This manual also contains a *Product Specification* which provides an overview of the characteristics and performance of the robot.

The *RAPID Reference Manual* contains a detailed explanation of the programming language as well as all *data types, instructions* and *functions*. They are described in alphabetical order for your convenience. If you are programming off-line, the *RAPID Reference Manual* will be particularly useful in this respect.

3 How to Read this Manual

Before you start reading through this manual, it is essential that you read *Chapter 3: Safety.* This tells you what you should or should not do to avoid injuring yourself or someone else.

Chapter 4: Basic Operation is an introduction to the basic operation and programming of the robot. It is recommended to be used as a tutorial, together with a robot or the PC software *QuickTeach*™.

You will find a general description of the robot, such as what happens on start-up or what the teach pendant does and looks like, in *Chapter 5: Starting up*.

Generally speaking, the robot is operated by means of different windows:

- Manual movement, see *Chapter 6: Jogging*.

- Manual operation of inputs and outputs, see *Chapter 7: Inputs and Outputs*.
- Programming and testing, see *Chapter 8: Programming and Testing*.
  The programming language is clearly described in *Chapter 9: The programming language RAPID*.
  For a more detailed description, see *RAPID Reference Manual*.
- Running production, see *Chapter 11: Production Running*.
- Setting system parameters, see *Chapter 12: System Parameters*.
- Copying programs, etc., see *Chapter 13: File Manager.*
- Service tools, see *Chapter 14: Service*.

Calibrating the robot, TCP and other coordinate systems, see *Chapter 10: Calibration.*

The program option Screen Viewer is described in *Chapter 15: ScreenViewer.*

Chapter 16 contains *Error Management* and *System and Error Messages.*

In *Chapter 17: Program Examples*, a number of programs are built up, step by step. Here you can learn a little about how to program, and also see the instructions in their correct context.

If you want to find out what a particular menu command does, you should refer to *Chapter 18: Quick Reference*. This chapter can also be used as a pocket guide when you are working with the robot.

If the robot is delivered or upgraded with some extra functionality this is described in *Chapter 19: Special Functionality in this Robot*.

To make things easier to locate and understand, *Chapter 20* contains an *index* and a *glossary.*

3.1 Typographic conventions

The commands located under any of the five menu keys at the top of the teach pendant display are written in the form of **Menu: Command**. For example, to activate the Print command in the File menu, you choose **File: Print**.

The names on the function keys and in the entry fields are specified in bold italic typeface, e.g. ***Modpos***.

Words belonging to the actual programming language, such as instruction names, are written in italics, e.g. *MoveL*.

Examples of programs are always displayed in the same way as they are output to diskette or a printer. This differs from what is displayed on the teach pendant in the following ways:

- Certain control words that are masked in the teach pendant display are printed, e.g. words indicating the start and end of a routine.
- Data and routine declarations are printed in the formal form, e.g. *VAR num reg1;*.

4 Reader’s Comments

You can use the next page to send us your comments about the manual. In this way, you will help us to improve the manual and make it easier for yourself to follow in the future. Thank you kindly for your cooperation.

CONTENTS

Page

1 General 3
1.1 Introduction 3
2 Applicable Safety Standards 3
3 Fire-Extinguishing 4
4 Definitions of Safety Functions 4
5 Safe Working Procedures 5
5.1 Normal operations 5
6 Programming, Testing and Servicing 5
7 Safety Functions 6
7.1 The safety control chain of operation 6
7.2 Emergency stops 7
7.3 Mode selection using the operating mode selector 7
7.4 Enabling device 8
7.5 Hold-to-run control 8
7.6 General Mode Safeguarded Stop (GS) connection 9
7.7 Automatic Mode Safeguarded Stop (AS) connection 10
7.8 Limiting the working space 10
7.9 Supplementary functions 10
8 Safety Risks Related to End Effectors 10
8.1 Gripper 10
8.2 Tools/workpieces 11
8.3 Pneumatic/hydraulic systems 11
9 Risks during Operation Disturbances 11
10 Risks during Installation and Service 11
11 Risks Associated with Live Electric Parts 12
12 Emergency Release of Mechanical Arm 13
13 Limitation of Liability 13
14 Related Information 13

Safety

1 General

This information on safety covers functions that have to do with the operation of the industrial robot.

The information does not cover how to design, install and operate a complete system, nor does it cover all peripheral equipment, which can influence the safety of the total system.

To protect personnel, the complete system has to be designed and installed in accordance with the safety requirements set forth in the standards and regulations of the country where the robot is installed.

The users of ABB industrial robots are responsible for ensuring that the applicable safety laws and regulations in the country concerned are observed and that the safety devices necessary to protect people working with the robot system have been designed and installed correctly.

People who work with robots must be familiar with the operation and handling of the industrial robot, described in applicable documents, e.g. Users's Guide and Product Manual.

The diskettes which contain the robot's control programs must not be changed in any way because this could lead to the deactivation of safety functions, such as reduced speed.

1.1 Introduction

Apart from the built-in safety functions, the robot is also supplied with an interface for the connection of external safety devices.

Via this interface, an external safety function can interact with other machines and peripheral equipment. This means that control signals can act on safety signals received from the peripheral equipment as well as from the robot.

In the Product Manual/*Installation*, instructions are provided for connecting safety devices between the robot and the peripheral equipment.

2 Applicable Safety Standards

The robot is designed in accordance with the requirements of ISO10218, Jan. 1992, Industrial Robot Safety. The robot also fulfils the ANSI/RIA 15.06-1992 stipulations.

3 Fire-Extinguishing

Use a CARBON DIOXIDE extinguisher in the event of a fire in the robot (manipulator or controller).

4 Definitions of Safety Functions

Emergency stop – IEC 204-1,10.7

A condition which overrides all other robot controls, removes drive power from robot axis actuators, stops all moving parts and removes power from other dangerous functions controlled by the robot.

Enabling device – ISO 11161, 3.4

A manually operated device which, when continuously activated in one position only, allows hazardous functions but does not initiate them. In any other position, hazardous functions can be stopped safely.

Safety stop – ISO 10218 (EN 775), 6.4.3

When a safety stop circuit is provided, each robot must be delivered with the necessary connections for the safeguards and interlocks associated with this circuit. It is necessary to reset the power to the machine actuators before any robot motion can be initiated. However, if only the power to the machine actuators is reset, this should not suffice to initiate any operation.

Reduced speed – ISO 10218 (EN 775), 3.2.17

A single, selectable velocity provided by the robot supplier which automatically restricts the robot velocity to that specified in order to allow sufficient time for people either to withdraw from the hazardous area or to stop the robot.

Interlock (for safeguarding) – ISO 10218 (EN 775), 3.2.8

A function that interconnects a guard(s) or a device(s) and the robot controller and/or power system of the robot and its associated equipment.

Hold-to-run control – ISO 10218 (EN 775), 3.2.7

A control which only allows movements during its manual actuation and which causes these movements to stop as soon as it is released.

5 Safe Working Procedures

Safe working procedures must be used to prevent injury. No safety device or circuit may be modified, bypassed or changed in any way, at any time.

5.1 Normal operations

All normal operations in automatic mode must be executed from outside the safeguarded space.

6 Programming, Testing and Servicing

The robot is extremely heavy and powerful, even at low speed. When entering into the robot's safeguarded space, the applicable safety regulations of the country concerned must be observed.

Operators must be aware of the fact that the robot can make unexpected movements. A pause (stop) in a pattern of movements may be followed by a movement at high speed. Operators must also be aware of the fact that external signals can affect robot programs in such a way that a certain pattern of movement changes without warning.

If work must be carried out within the robot's work envelope, the following points must be observed:

- The operating mode selector on the controller must be in the manual mode position to render the enabling device operative and to block operation from a computer link or remote control panel.
- The robot's speed is limited to max. 250 mm/s (10 inches/s) when the operating mode selector is in position < 250 mm/s. This should be the normal position when entering the working space. The position 100% – full speed – may only be used by trained personnel who are aware of the risks that this entails.

Do not change "Transm gear ratio" or other kinematic parameters from the teach pendant or a PC. This will affect the safety function *Reduced speed 250 mm/s*.

- During programming and testing, the enabling device must be released as soon as there is no need for the robot to move.

The enabling device must never be rendered inoperative in any way.

- The programmer must always take the teach pendant with him/her when entering through the safety gate to the robot's working space so that no-one else can take over control of the robot without his/her knowledge.

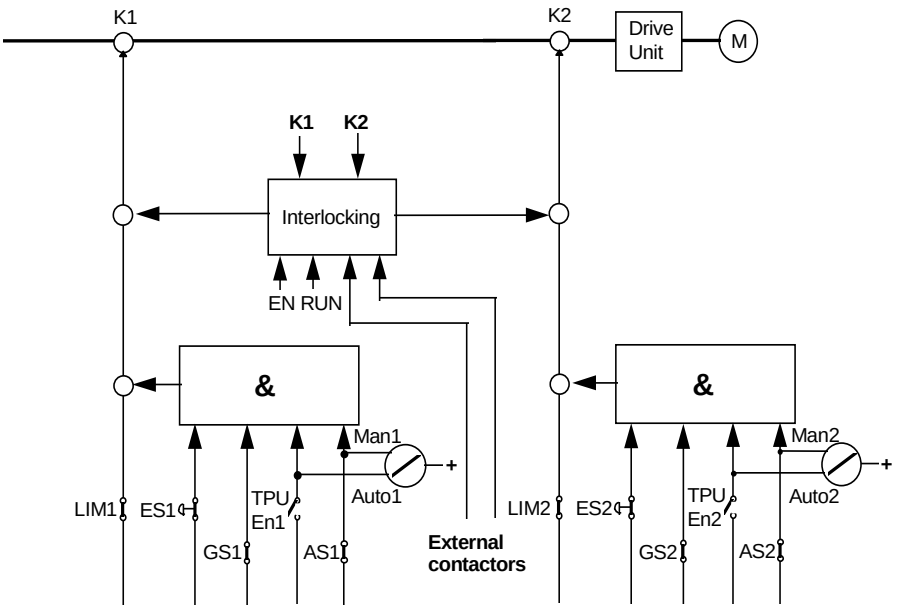
7 Safety Functions

7.1 The safety control chain of operation

The safety control chain of operation is based on dual electrical safety chains which interact with the robot computer and enable the MOTORS ON mode.

Each electrical safety chain consist of several switches connected in such a way that all of them must be closed before the robot can be set to MOTORS ON mode. MOTORS ON mode means that drive power is supplied to the motors.

If any contact in the safety chain of operation is open, the robot always reverts to MOTORS OFF mode. MOTORS OFF mode means that drive power is removed from the robot's motors and the brakes are applied.



The status of the switches is indicated by LEDs on top of the panel module in the control cabinet and is also displayed on the teach pendant (I/O window).

After a stop, the switch must be reset at the unit which caused the stop before the robot can be ordered to start again.

The time limits for the central two channel cyclic supervisions of the safety control chain is between 2 and 4 second.

The safety chains must never be bypassed, modified or changed in any other way.

7.2 Emergency stops

An emergency stop should be activated if there is a danger to people or equipment. Built-in emergency stop buttons are located on the operator's panel of the robot controller and on the teach pendant.

External emergency stop devices (buttons, etc.) can be connected to the safety chain by the user (see Product Manual/*Installation*). They must be connected in accordance with the applicable standards for emergency stop circuits.

Before commissioning the robot, all emergency stop buttons or other safety equipment must be checked by the user to ensure their proper operation.

Before switching to MOTORS ON mode again, establish the reason for the stop and rectify the fault.

7.3 Mode selection using the operating mode selector

The applicable safety requirements for using robots, laid down in accordance with ISO/DIS 10218, are characterised by different modes, selected by means of control devices and with clear-cut positions.

One automatic and two manual modes are available:

Manual mode:
< 250 mm/s - max. speed is 250mm/s
100% - full speed

Automatic mode: The robot can be operated via a remote control device

The manual mode, < 250 mm/s or 100%, must be selected whenever anyone enters the robot's safeguarded space. The robot must be operated using the teach pendant and, if 100% is selected, using Hold-to-run control.

In automatic mode, the operating mode selector is switched to , and all safety arrangements, such as doors, gates, light curtains, light beams and sensitive mats, etc., are active. No-one may enter the robot's safeguarded space. All controls, such as emergency stops, the control panel and control cabinet, must be easily accessible from outside the safeguarded space.

Programming and testing at reduced speed

Robot movements at reduced speed can be carried out as follows:

- Set the operating mode selector to <250 mm/s
- Programs can only be started using the teach pendant with the enabling device activated.

The automatic mode safeguarded space stop (AS) function is not active in this mode.

Testing at full speed

Robot movements at programmed speed can be carried out as follows:

- Set the operating mode selector to 100%
- Programs can only be started using the teach pendant with the enabling device activated.

For “Hold-to-run control”, the Hold-to-run button must be activated. Releasing the button stops program execution.

The 100% mode may only be used by trained personnel. The applicable laws and regulations of the countries where the robot is used must always be observed.

Automatic operation

Automatic operation may start when the following conditions are fulfilled:

- The operating mode selector is set to
- The MOTORS ON mode is selected

Either the teach pendant can be used to start the program or a connected remote control device. These functions should be wired and interlocked in accordance with the applicable safety instructions and the operator must always be outside the safeguarded space.

7.4 Enabling device

When the operating mode selector is in the MANUAL or MANUAL FULL SPEED position, the robot can be set to the MOTORS ON mode by depressing the enabling device on the teach pendant.

Should the robot revert to the MOTORS OFF mode for any reason while the enabling device is depressed, the latter must be released before the robot can be returned to the MOTORS ON mode again. This is a safety function designed to prevent the enabling device from being rendered inactive.

When the enabling device is released, the drive power to the motors is switched off, the brakes are applied and the robot reverts to the MOTORS OFF mode.

If the enabling device is reactivated, the robot changes to the MOTORS ON mode.

7.5 Hold-to-run control

This function is always active when the operating mode selector is in the MANUAL FULL SPEED position. It is possible to set a parameter to make this function active also when the operating mode selector is in the MANUAL position.

When the Hold-to-run control is active, the enabling device and the Hold-to-run button on the teach pendant must be depressed in order to execute a program. When the button is released, the axis (axes) movements stop and the robot remains in the MOTORS ON mode.

Here is a detailed description of how to execute a program in Hold-to-run control:

• Activate the enabling device on the teach pendant.

• Choose execution mode using the function keys on the teach pendant:

- **Start** (continuous running of the program)
- **FWD** (one instruction forwards)
- **BWD** (one instruction backwards)

• Wait for the Hold-to-run alert box.

• Activate the Hold-to-run button on the teach pendant.

Now the program will run (with the chosen execution mode) as long as the Hold-to-run button is pressed. Releasing the button stops program execution and activating the button will start program execution again.

For FWD and BWD execution modes, the next instruction is run by releasing and activating the Hold-to-run button.

It is possible to change execution mode when the Hold-to-run button is released and then continue the program execution with the new execution mode, by just activating the Hold-to-run button again, i.e. no alert box is shown.

If the program execution was stopped with the Stop button on the teach pendant, the program execution will be continued by releasing and activating the Hold-to-run button.

When the enabling device on the teach pendant is released, the sequence described above must be repeated from the beginning.

7.6 General Mode Safeguarded Stop (GS) connection

The GS connection is provided for interlocking external safety devices, such as light curtains, light beams or sensitive mats. The GS is active regardless of the position of the operating mode selector.

When this connection is open the robot changes to the MOTORS OFF mode. To reset to MOTORS ON mode, the device that initiated the safety stop must be interlocked in accordance with applicable safety regulations. This is not normally done by resetting the device itself.

7.7 Automatic Mode Safeguarded Stop (AS) connection

The AS connection is provided for interlocking external safety devices, such as light curtains, light beams or sensitive mats used externally by the system builder. The AS is especially intended for use in automatic mode, during normal program execution.

The AS is by-passed when the operating mode selector is in the MANUAL or MANUAL FULL SPEED position.

7.8 Limiting the working space

For certain applications, movement about the robot's main axes must be limited in order to create a sufficiently large safety zone. This will reduce the risk of damage to the robot if it collides with external safety arrangements, such as barriers, etc.

Movement about axes 1, 2 and 3 can be limited with adjustable mechanical stops or by means of electrical limit switches. If the working space is limited by means of stops or switches, the corresponding software limitation parameters must also be changed. If necessary, movement of the three wrist axes can also be limited by the computer software. Limitation of movement of the axes must be carried out by the user.

7.9 Supplementary functions

Functions via specific digital inputs:

- A stop can be activated via a connection with a digital input. Digital inputs can be used to stop programs if, for example, a fault occurs in the peripheral equipment.

Functions via specific digital outputs:

- Error – indicates a fault in the robot system.
- Cycle_on – indicates that the robot is executing a program.
- MotOnState/MotOffState – indicates that the robot is in MOTORS ON / MOTORS OFF mode.
- EmStop - indicates that the robot is in emergency stop state.
- AutoOn - indicates that the robot is in automatic mode.

8 Safety Risks Related to End Effectors

8.1 Gripper

If a gripper is used to hold a workpiece, inadvertent loosening of the workpiece must be prevented.

8.2 Tools/workpieces

It must be possible to turn off tools, such as milling cutters, etc., safely. Make sure that guards remain closed until the cutters stop rotating.

Grippers must be designed so that they retain workpieces in the event of a power failure or a disturbance of the controller. It should be possible to release parts by manual operation (valves).

8.3 Pneumatic/hydraulic systems

Special safety regulations apply to pneumatic and hydraulic systems.

Residual energy may be present in these systems so, after shutdown, particular care must be taken.

The pressure in pneumatic and hydraulic systems must be released before starting to repair them. Gravity may cause any parts or objects held by these systems to drop. Dump valves should be used in case of emergency. Shot bolts should be used to prevent tools, etc., from falling due to gravity.

9 Risks during Operation Disturbances

If the working process is interrupted, extra care must be taken due to risks other than those associated with regular operation. Such an interruption may have to be rectified manually.

Remedial action must only ever be carried out by trained personnel who are familiar with the entire installation as well as the special risks associated with its different parts.

The industrial robot is a flexible tool which can be used in many different industrial applications. All work must be carried out professionally and in accordance with applicable safety regulations. Care must be taken at all times.

10 Risks during Installation and Service

To prevent injuries and damage during the installation of the robot system, the regulations applicable in the country concerned and the instructions of ABB Robotics must be complied with. Special attention must be paid to the following points:

- The supplier of the complete system must ensure that all circuits used in the safety function are interlocked in accordance with the applicable standards for that function.
- The instructions in the Product Manual/*Installation* must always be followed.
- The mains supply to the robot must be connected in such a way that it can be turned off outside the robot’s working space.

- The supplier of the complete system must ensure that all circuits used in the emergency stop function are interlocked in a safe manner, in accordance with the applicable standards for the emergency stop function.
- Emergency stop buttons must be positioned in easily accessible places so that the robot can be stopped quickly.
- Safety zones, which have to be crossed before admittance, must be set up in front of the robot's working space. Light beams or sensitive mats are suitable devices.
- Turntables or the like should be used to keep the operator away from the robot's working space.
- Those in charge of operations must make sure that safety instructions are available for the installation in question.
- Those who install the robot must have the appropriate training for the robot system in question and in any safety matters associated with it.

Although troubleshooting may, on occasion, have to be carried out while the power supply is turned on, the robot must be turned off (by setting the mains switch to OFF) when repairing faults, disconnecting electric leads and disconnecting or connecting units.

Even if the power supply for the robot is turned off, you can still injure yourself.

- The axes are affected by the force of gravity when the brakes are released. In addition to the risk of being hit by moving robot parts, you run the risk of being crushed by the tie rod.
- Energy, stored in the robot for the purpose of counterbalancing certain axes, may be released if the robot, or parts thereof, is dismantled.
- When dismantling/assembling mechanical units, watch out for falling objects.
- Be aware of stored energy (DC link) and hot parts in the controller.
- Units inside the controller, e.g. I/O modules, can be supplied with external power.

11 Risks Associated with Live Electric Parts

Controller

A danger of high voltage is associated with the following parts:

- The mains supply/mains switch
- The power unit
- The power supply unit for the computer system (55 V AC)
- The rectifier unit (260 V AC and 370 V DC. NB: Capacitors!)
- The drive unit (370 V DC)
- The service outlets (115/230 VAC)
- The power supply unit for tools, or special power supply units for the machining process

- The external voltage connected to the control cabinet remains live even when the robot is disconnected from the mains.
- Additional connections

Manipulator

A danger of high voltage is associated with the manipulator in:

- The power supply for the motors (up to 370 V DC)
- The user connections for tools or other parts of the installation (see *Installation*, max. 230 V AC)

Tools, material handling devices, etc.

Tools, material handling devices, etc., may be live even if the robot system is in the OFF position. Power supply cables which are in motion during the working process may be damaged.

12 Emergency Release of Mechanical Arm

If an emergency situation occur where a person is caught by the mechanical robot arm, the brake release buttons should be pressed whereby the arms can be moved to release the person. To move the arms by manpower is normally possible on the smaller robots (1400 and 2400), but for the bigger ones it might not be possible without a mechanical lifting device, like an overhead crane.

If power is not available the brakes are applied, and therefore manpower might not be sufficient for any robot.

Before releasing the brakes, secure that the weight of the arms not enhance the press force on the caught person.

13 Limitation of Liability

The above information regarding safety must not be construed as a warranty by ABB Robotics that the industrial robot will not cause injury or damage even if all safety instructions have been complied with.

14 Related Information

	Described in:
Installation of safety devices	Product Manual - *Installation and Commissioning*
Changing robot modes	User’s Guide - *Starting up*
Limiting the working space	Product Manual - *Installation and Commissioning*

CONTENTS

Page

1 Introduction **3**
2 Safety **5**
3 System Overview **7**
3.1 General 7
3.2 The manipulator 8
3.3 The controller 8
3.4 Operator's panel 9
3.5 Teach pendant 10
4 Starting the System **15**
5 Working with Windows **17**
6 Jogging the Robot Using the Joystick **21**
6.1 Linear jogging 21
6.2 Fine positioning 23
7 Selecting a Program **25**
7.1 Using the training program 25
8 Starting the Program **31**
9 Stopping the Program **35**
10 Automatic Mode **37**
11 Errors **39**
12 Switching the robot off **41**
13 Changing a Program **43**
13.1 Modifying positions 43
13.2 Changing arguments 45
13.3 Adding instructions 46
13.4 Programming a delay 48
14 Storing the Program on Diskette **51**
14.1 Storing on diskette 51
15 Printing Programs **55**
15.1 Using a PC 55
16 I/O Signals **57**
16.1 Programming an I/O instruction 57

1 Introduction

This manual explains the basics of handling and operating an ABB robot. You do not need any previous experience of robots to understand its contents.

The manual is divided into chapters, each of which describes a particular work task and how to go about performing it. The chapters complement one another and should, therefore, be read in the order they appear in the book.

It is an advantage if you have access to a robot (or the PC-program Quick Teach) when you use this manual, but just reading it should help you understand the basic operation of a robot.

The manual is written to suit a standard installation. Differences can therefore occur, depending on the configuration of the system.

Please note that this manual describes only one method of carrying out any of the normal work tasks and, if you are an experienced user, there may be other methods. For other methods and more detailed information, see the following manuals.

The **User's Guide** is a reference manual with step by step instructions on how to perform various tasks.

The programming language is described in the **RAPID Reference Manual**.

The **Product Manual** describes how to install the robot, as well as maintenance procedures and troubleshooting.

If you just wish to be able to start programs, run the robot with the joystick, load programs from diskette, etc., it is not necessary to read Chapters 14-16 in this manual.

2 Safety

Operational procedures, during training or at any other time, must be carried out safely.

Entering the safeguarded space around the robot may cause severe injury and should be avoided whenever possible. However, if this is necessary, then only authorised personnel may enter the area. The existing safety regulations must always be taken into consideration.

The safety regulations are specified in the chapters on safety in the User's Guide and in relevant plant documentation (if any).

3 System Overview

3.1 General

A robot is made up of two principal parts:

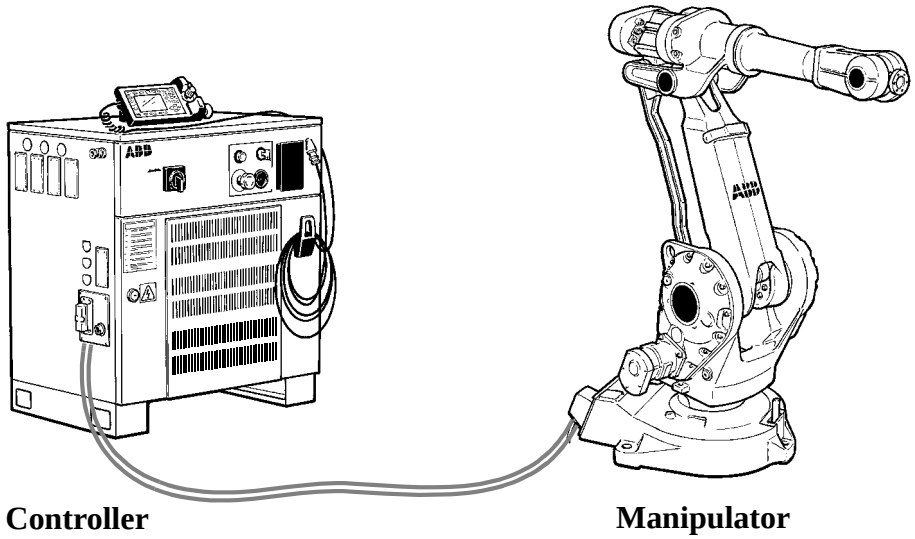


Figure 1 The controller and manipulator are connected by two cables.

You can communicate with the robot using a teach pendant and/or an operator's panel, located on the controller (see Figure 2).

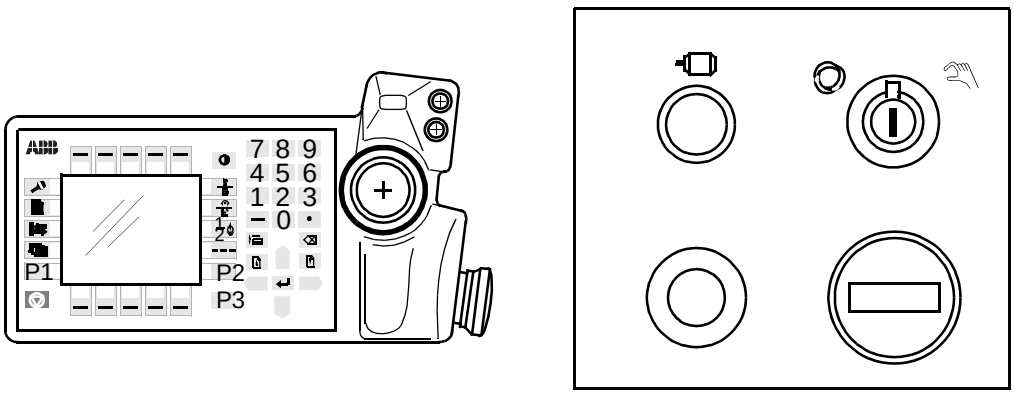


Figure 2 The teach pendant and the operator's panel.

3.2 The manipulator

Figure 3 shows the directions in which the various axes of the manipulator can move and what these are called.

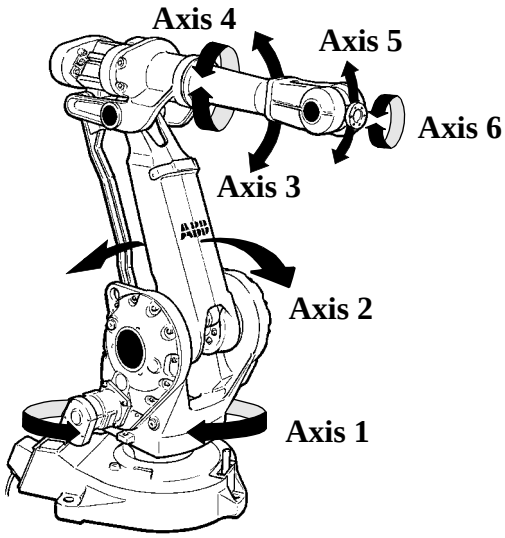


Figure 3 Manipulator, IRB 2400.

3.3 The controller

Figure 4 illustrates the principal parts of the controller.

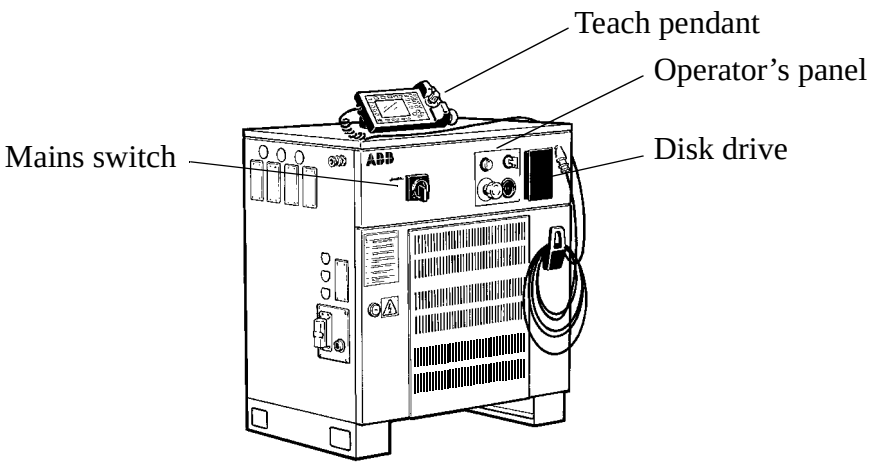


Figure 4 The S4C control system.

3.4 Operator's panel

Figure 5 below shows a close-up of the operator's panel. A short explanation of the push buttons follows.

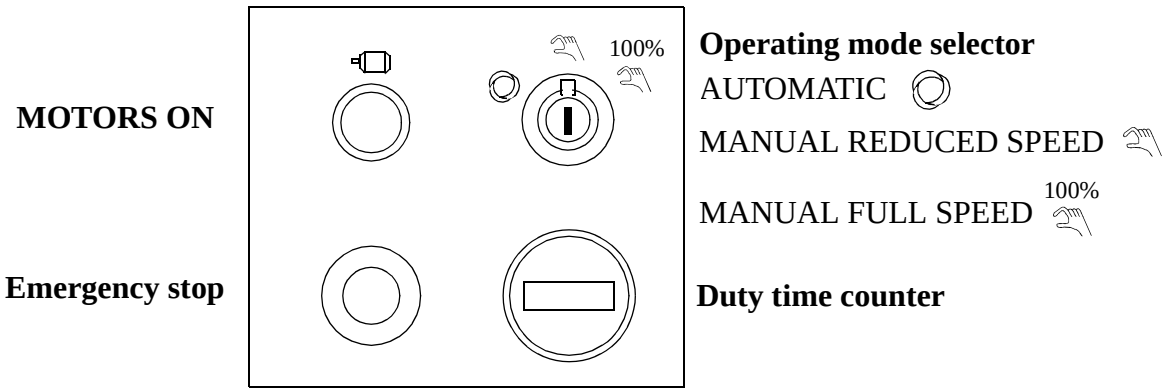


Figure 5 The operator's panel.

MOTORS ON

In the **MOTORS ON** state, the motors of the robot are activated and the **MOTORS ON** button is continuously lit.

Operating mode AUTOMATIC (Production mode)

Used when running ready-made programs in production. It is not possible to move the robot with the joystick in this mode.

Operating mode MANUAL REDUCED SPEED (Programming mode)

Used when working inside the robot's working area and when programming the robot. Also used to set the robot in **MOTORS OFF** state.

Operating mode MANUAL FULL SPEED (Testing mode, Option not standard)

Used to test run the robot program at full programming speed.

Emergency stop

The robot stops – regardless of which state or mode the system is in – immediately when the emergency stop button is pressed. The button remains pressed in and, to turn to **MOTORS ON** again, must be returned to its original position.

Duty time counter

Indicates the operating time for the manipulator (released brakes).

3.5 Teach pendant

The teach pendant is described briefly below; see Figure 6 and Figure 7.

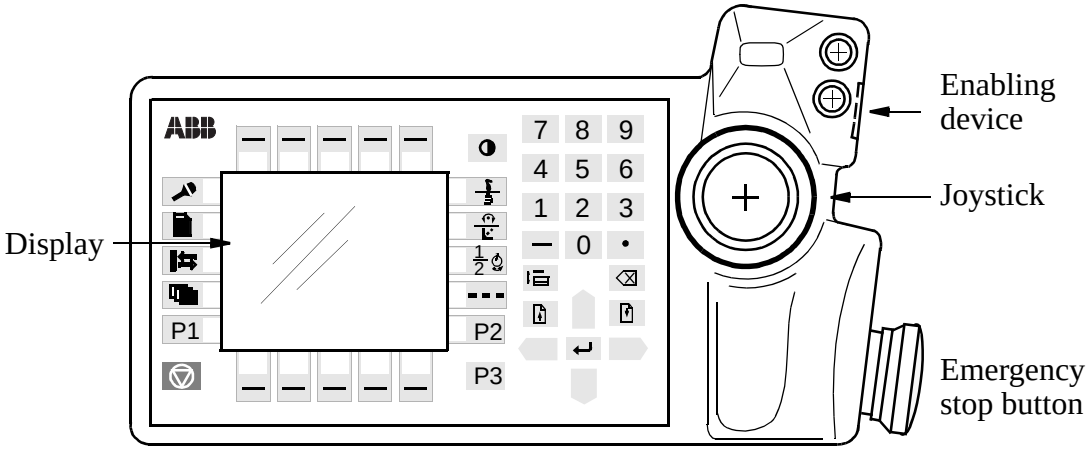


Figure 6 The teach pendant.

Emergency stop

The robot stops – regardless of which state or mode the system is in – immediately the emergency stop button is pressed. The button remains pressed in and, to turn to **MOTORS ON** again, must be returned to its original position.

Enabling device (for safe operation)

A push button on the teach pendant which, when pressed halfway in, takes the system to **MOTORS ON** (if the operating mode selector is switched to one of the two manual modes). When the enabling device is released or pushed all the way in, the robot is taken to the **MOTORS OFF** state.

If the enabling device is released and pressed in halfway again within half a second, the robot will not return to the **MOTORS ON** state.
If this happens, the enabling device must first be released and then pushed halfway in again.

The enabling device should only be activated when the robot is to be moved – either with the joystick or during program execution.

Joystick

The joystick is used to jog (move) the robot manually; e.g. when programming the robot.

Display

Used to display all information during programming, to change programs, etc. It can accommodate 16 lines; each line can accommodate 40 characters.

Figure 7 shows the names of the various keys on the teach pendant.

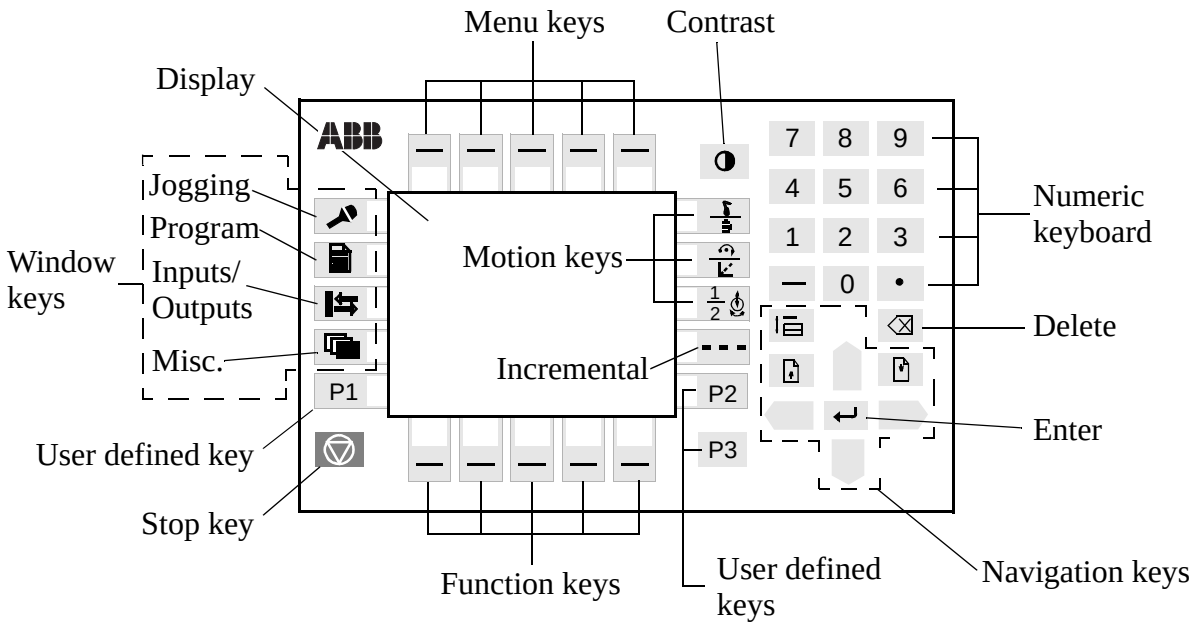
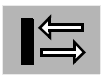


Figure 7 An overview of the various keys on the teach pendant, Version 2.

Window keys (to select a window to work with on the display):

Jogging: Used to jog the robot.

Program: Used to program and test.



Inputs/Outputs: Used to manually operate the input and output signals connected to the robot.

Misc.: Miscellaneous; other windows, i.e. the System Parameters, Service, Production and File Manager windows.

Navigation keys (to move the cursor within a window on the display):

List: Press to move the cursor from one part of the window to another (normally separated by a double line).

Previous/Next page: Press to see the next/previous page.

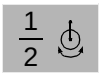
Up and Down arrows: Press to move the cursor up or down.

Left and Right arrows: Press to move the cursor to the left or right.

Motion keys: (to select how the robot or other peripheral equipment should move when using the joystick – during manual operation):

Motion Unit: Press to jog the robot or other mechanical units.

Motion Type: Press to select how the robot should be jogged, reorientation or linear.



Motion Type: Axis by axis movement. 1 = axis 1-3, 2 = axis 4-6

Incremental: Incremental jogging on/off

Other keys:

Stop: Stops program execution.

Contrast: Adjusts contrast of the display

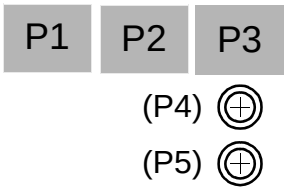
Menu keys: Press to display menus containing various commands.

Function keys: Press to select the various commands directly.

Delete: Deletes the data selected on the display.

Enter: Press to input data.

Programmable keys:



Functions to be defined by the user.

4 Starting the System

You are now going to turn the system on, i.e. get it ready for programming, running programs, etc.

Before you switch the system on, make sure that no-one is inside the safeguarded space around the robot.

1. Switch the mains switch on (see Figure 8). The robot is then automatically checked.

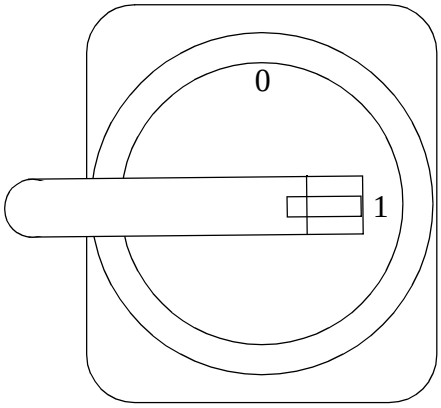


Figure 8 Mains switch

After the system has been checked and no errors are located, the following message (see Figure 9) appears on the display.

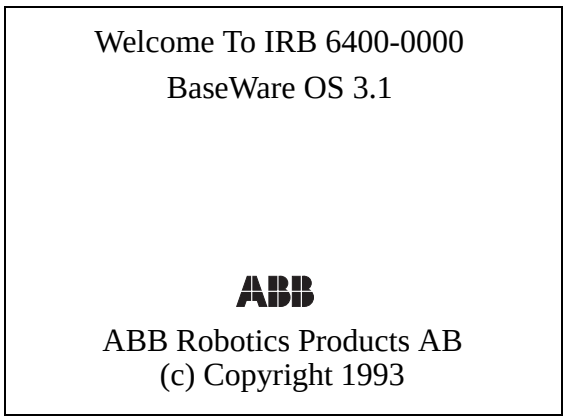


Figure 9 The “welcome” window may vary slightly depending on the type and version of your robot.

5 Working with Windows

In this chapter, you will find out about the basics of working with windows. The following example shows the window for **Inputs/Outputs** (manual handling of in- and outputs).

1. Press the **Inputs/Outputs** window key (see Figure 10.)

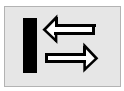


Figure 10 The Inputs/Outputs application key, both versions.

2. The window for manual I/O is now shown on the display, as in Figure 11. The appearance of the I/O list may vary depending on how the signals have been defined and how many I/O boards there are in the system.

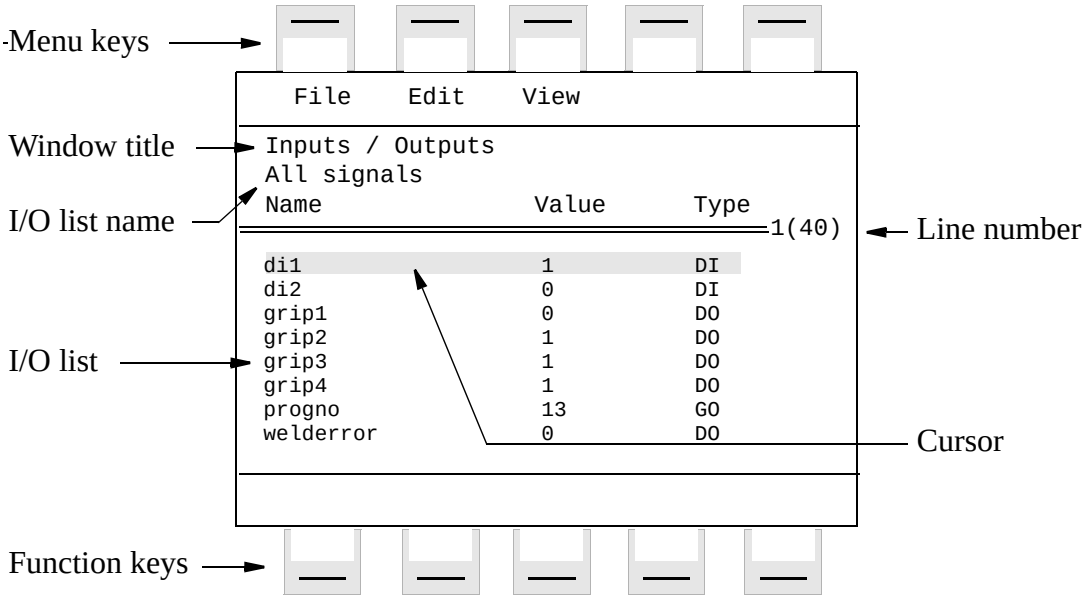


Figure 11 Window for manual I/O handling.

When a digital output is selected, its status can be changed using the function keys.

3. You can select a signal in the list (move the cursor) in several ways:

Movement	**Select**
One line up	ArrowUp
One line down	ArrowDown
To the first line in the list	**Goto top** from the **Edit** menu
To the last line in the list	**Goto bottom** from the **Edit** menu
To the next page	NextPage
To the previous page	PreviousPage
To select a specific line in the list	**Goto** from the **Edit** menu; enter the desired line number and press ***OK***

4. Windows are sometimes divided in two by a double line (see Figure 12).
5. When an output is selected, two function keys will be displayed (see Figure 12).

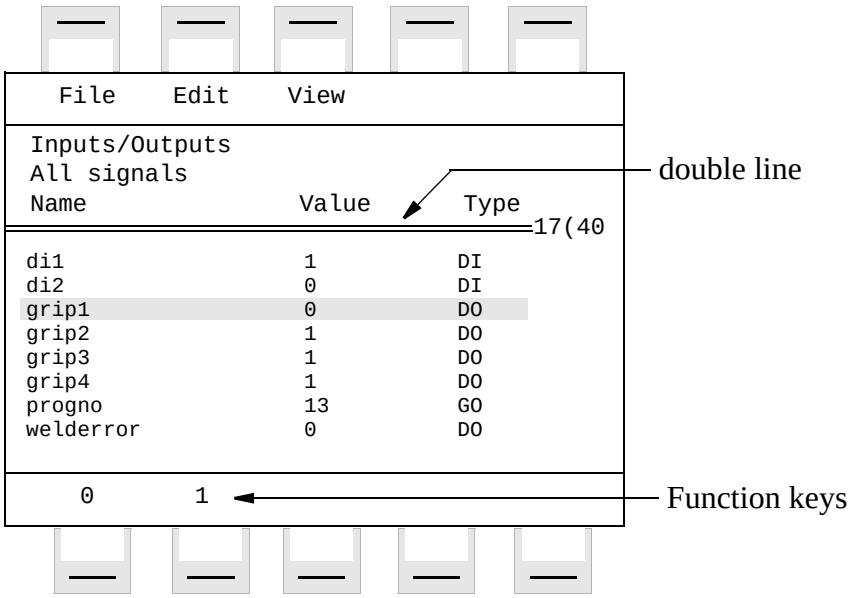


Figure 12 A window with two parts.

In some windows, you can move the cursor between the different parts of the window.

In these windows, move the cursor using the List key .

6. There are four window keys on the teach pendant (see Figure 13 below and Chapter 3).

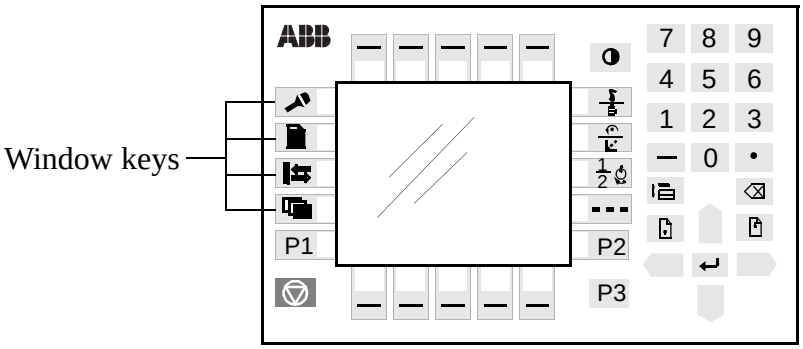


Figure 13 The four window keys.

When you press a window key, the active window will be hidden under the new one. Each time you select a window, it will look the same as it did the last time you worked with it.

6 Jogging the Robot Using the Joystick

You can move (jog) the robot using the joystick on the teach pendant. This chapter describes how to jog the robot linearly (i.e. in a straight line) and step by step, to make it easier to position the robot exactly (known as incremental jogging).

6.1 Linear jogging

1. Make sure that the operating mode selector is in the < 250 mm/s position , as shown in Figure 14.

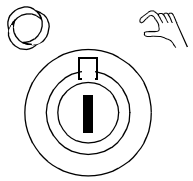


Figure 14 The maximum speed during manual operation is 250 mm/s.

2. Check that the **Robot** motion unit and the **Linear** motion type are selected (see Figure 15).

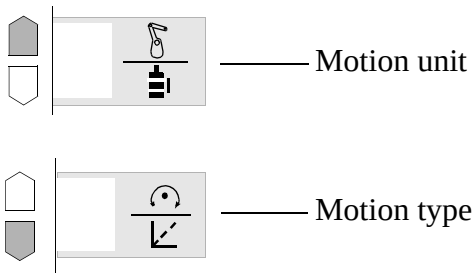


Figure 15 Motion keys, LEDs shows the current settings.

With the **Motion unit** key, you can choose between operating the robot, or some other unit connected to the controller, using the joystick. Select the robot for this exercise.

With the **Motion type** key, you can choose the way you want the robot to move when you use the joystick during manual operation.

You can choose:

- linear movement
- reorientation of a particular end-effector
- axis-by-axis movement (group 1: axes 1-3; group 2: axes 4-6)

We will use linear motion for the purposes of this exercise.

When linear type motion is selected, the robot will move as shown in Figure 16.

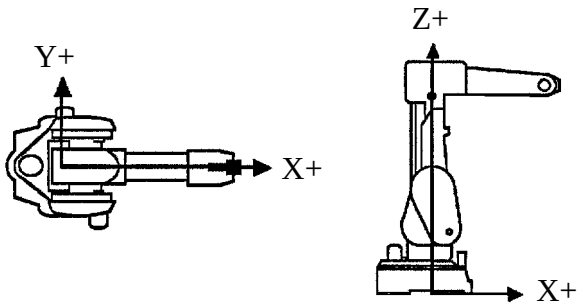


Figure 16 X, Y, Z form the robot's coordinate system.

The point that will move linearly, along the axes of the coordinate system above, is called the **Tool Centre Point (TCP) 0**. It is located at the front of the upper arm, in the centre of the robot's face plate (see Figure 17).

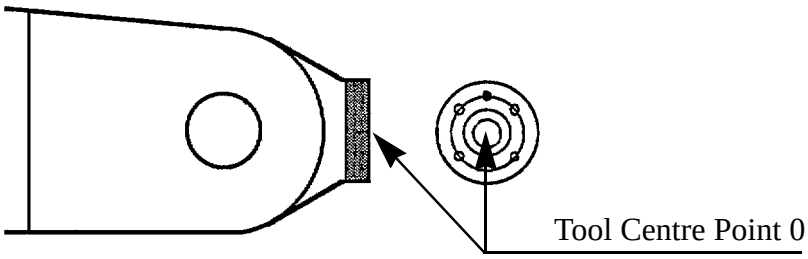


Figure 17 The centre of the face plate is called TCP 0.

3. Push the enabling device halfway in to switch the **MOTORS ON**.

4. Now, jog the robot using the joystick.
   Standing in front of the robot, the TCP 0 will, depending on how you move the joystick, move linearly along the X-, Y- and Z-axes (see Figure 18).

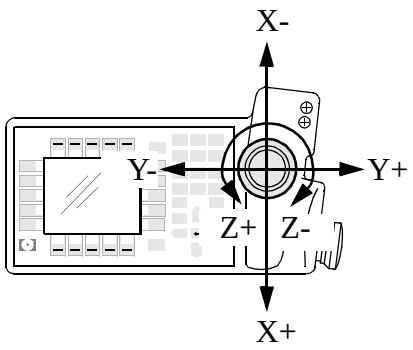


Figure 18 Robot movements with different joystick deflections.

Try jogging the robot in the directions corresponding to X, Y and Z above.
You can also combine the various movements of the joystick and move in several directions simultaneously. Note that the speed of the robot depends on how much you move the joystick. The larger deflection, the faster the robot moves.

6.2 Fine positioning

1. Press the **Jogging** window key (see Figure 19).

Figure 19 The Jogging window key.

A window like the one in Figure 20 will appear.

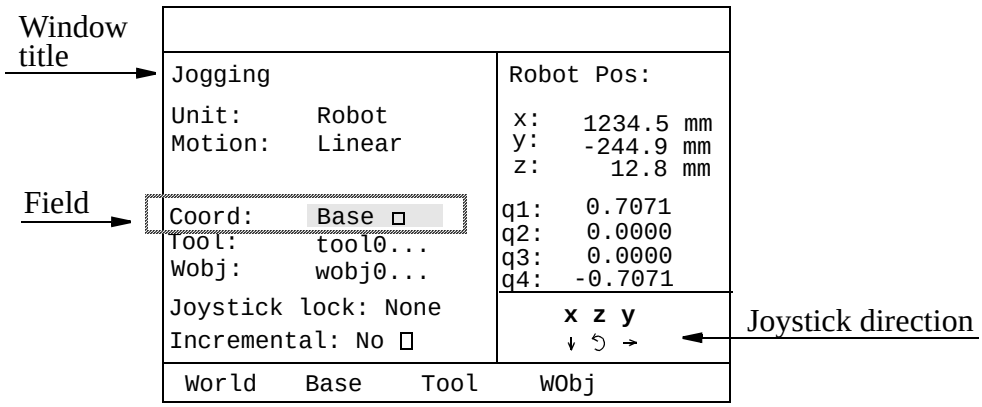


Figure 20 The Jogging window.

The appearance of the window changes depending on the type of window selected (i.e. depending on what you want to do).

The **Menu keys** perform different commands. The list of commands available is displayed in a pull-down menu when you press any of the menu keys.

The area enclosed by a dashed line is called a **Field**. The highlighted (shaded, grey) area is known as an input field and can be changed by selecting a different function using one of the **Function keys** (or, in some cases, using the **Motion keys** on the teach pendant).

The highlighted input field in Figure 20 is marked with a " □ " after it which means that:

- Selection is done using a **Function key**

2. Move the cursor (the shaded field) to the **Incremental** field using the **Down arrow** key on the teach pendant (see Figure 21).

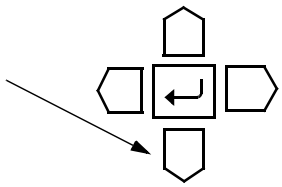


Figure 21 The Down arrow navigation key.

3. If you move the cursor to the **Incremental** field, as in Figure 22, you can choose incremental jogging by pressing one the function keys.

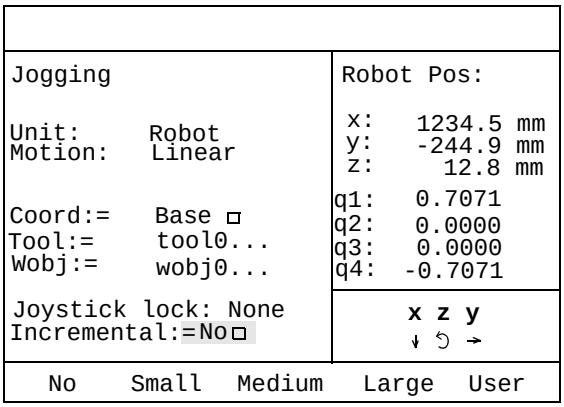


Figure 22 Selection of incremental jogging.

If you press the ***Small, Medium***, ***Large*** or ***User*** function key, the No in the Incremental field will be immediately replaced.

The robot will then move one step at a time each time you move the joystick; the size of the steps (Small, Medium, Large or User defined) will depend on your choice.

You can also use the key --- to turn incremental movement on and off.

Try operating the robot using the joystick and note how the robot moves.

More information on manual operation, the various coordinate systems, etc., can be found in the chapter on jogging in the User's Guide.

If you do not wish to continue this exercise, switch off the system as explained in Chapter 12 Switching the robot off.

7 Selecting a Program

This chapter explains how to open (choose) a program. A **program** is usually made up of three different parts, one **main routine** (always present), a number of **subroutines** and **program data**. Only one main routine is permitted per program (see Figure 23).

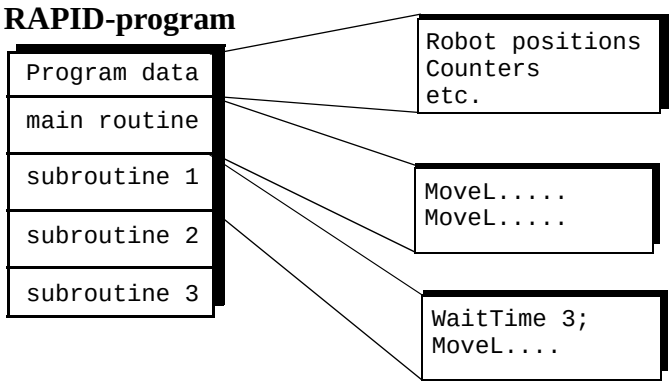


Figure 23 Program structure.

If you open a program, that program replaces the program in the robot's memory. When a program is opened, the main routine will be shown on the display with the first instruction in the main routine selected (highlighted).

7.1 Using the training program

The training program is stored on the system diskette "Controller parameters", under the directory DEMO, and is called "EXERCISE".

1. Turn the operating mode selector on the operator's panel to < 250 mm/s.
2. Press the **Program** window key (see Figure 24).

Figure 24 The Program window key.

If there is no program in the robot's memory, the following window will appear (Figure 25), otherwise you will see the program that is stored in the memory of the robot.

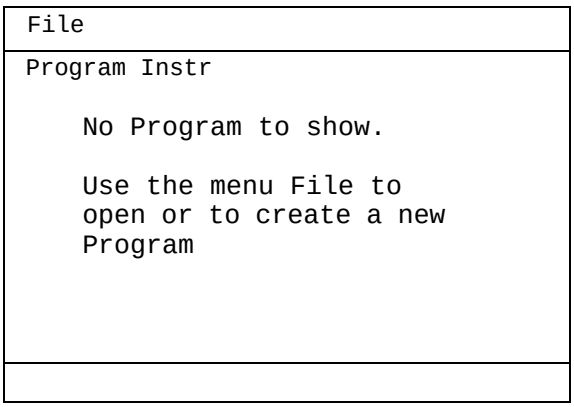


Figure 25 The Program window.

3. Insert the Setup diskette into the disk drive at the front of the cabinet. The diskette should be inserted as in Figure 26.

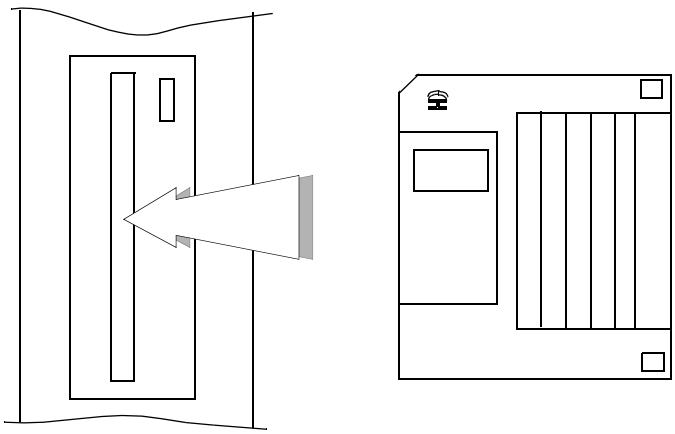


Figure 26 Inserting the diskette.

4. Press the **File** menu key (see Figure 25 above). The window in Figure 27 will appear.

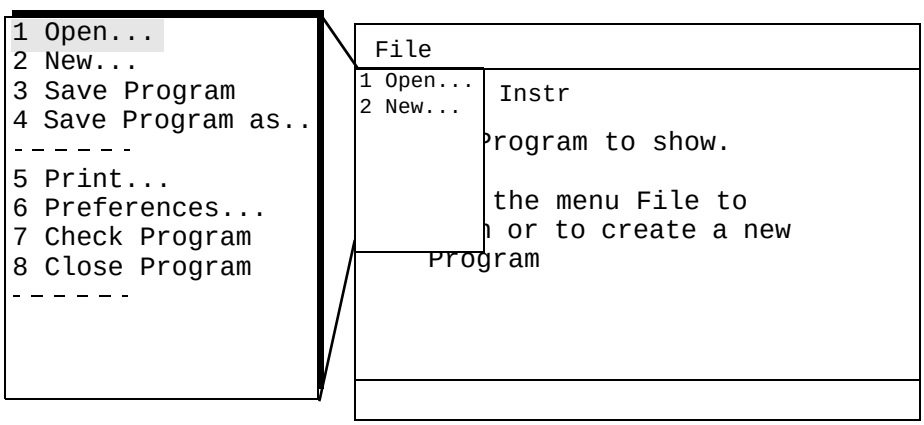


Figure 27 The File menu when there is no program in the robot's internal memory.

The menu that now appears on your display is called a “pull-down menu”. All commands that can be chosen from the **File** menu are listed here (commands that cannot be chosen are indicated by parentheses). The other menu keys work in the same way.

From now on we will write **File: Open...**, **File: Save**, etc. The name on the left of the colon is the name of the menu and the name on the right stands for the command you should choose.

The first function in the pull-down menu is always highlighted when you press the menu key. You can move the cursor within the menu using the arrow keys on the teach pendant (see Figure 28). When you have selected the command you want to choose, press **Enter**.

You can also use the numeric keyboard to choose a command; to do this enter the number in front of the command.

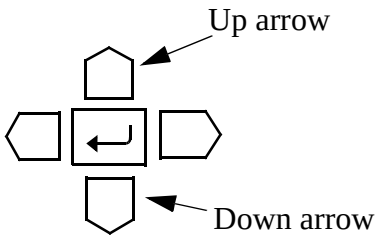


Figure 28 Navigation keys: Up arrow and Down arrow.

Three dots “...” following the command means that a dialog box will appear when that command is chosen.

To remove a pull-down menu, press the menu key with which you opened it.

5. After selecting **Open...**, press Enter (see Figure 29).
This means that the “Open...” command will be carried out. However, as it has three dots “...” after it, the command will not be performed directly since more information is required. You must now, in this case, choose the particular program you wish to open.

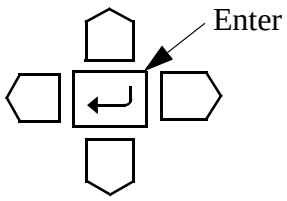


Figure 29 The Enter key on the teach pendant.

Using the ***Unit*** function key, you can switch between the robot's internal memory (ram1disk), the diskette unit (flp1:) or some other type of mass storage device.

6. Press ***Unit***. Check that "flp1:" appears after Massmemory unit:=. A dialog box will appear and the contents of the diskette will be shown, as in Figure 30.

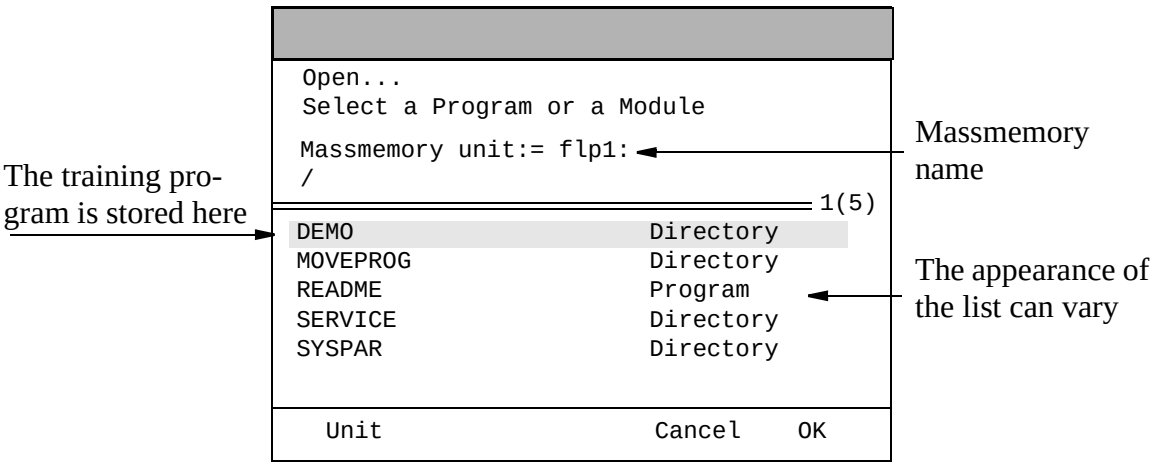


Figure 30 The Open dialog box.

If a dialog box (does not have any menus) is closed by pressing ***Cancel***, the command requested will not be carried out. If you press ***OK***, the command requested will be carried out and the dialog box will close.

7. Select **DEMO**. Move the cursor with the Up and Down arrow keys.
8. Press Enter ↵ .
9. Select **EXERCISE**.
10. Press ***OK*** and a window like the one in Figure 31 will appear.

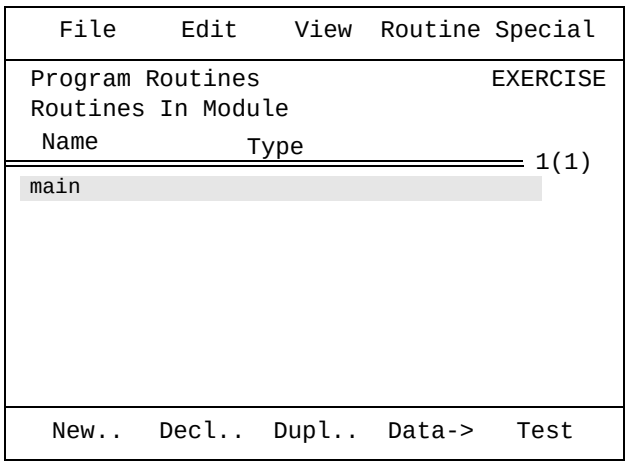


Figure 31 Opening the main file of the training program.

11. Then press Enter [↵] . The window in Figure 32 appears.

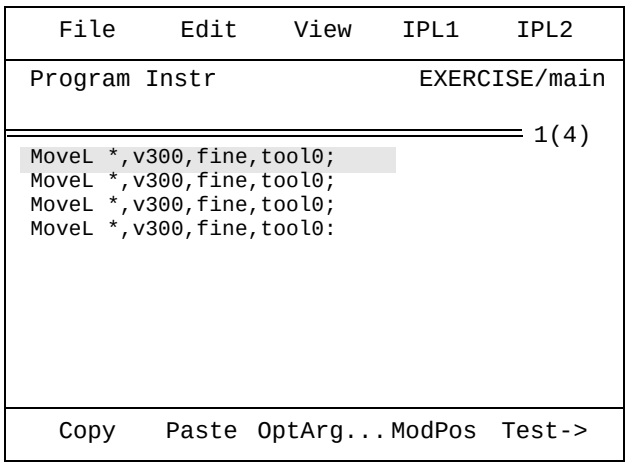


Figure 32 The training program appears on the display.

This is the main routine in the training program. It consists of four “move” instructions.

The routines consist of different types of instructions, such as move instructions, wait instructions, etc. Each instruction is followed by different arguments. Arguments can, depending on their type, be changed or omitted altogether. Figure 33 indicates an example of an instruction.

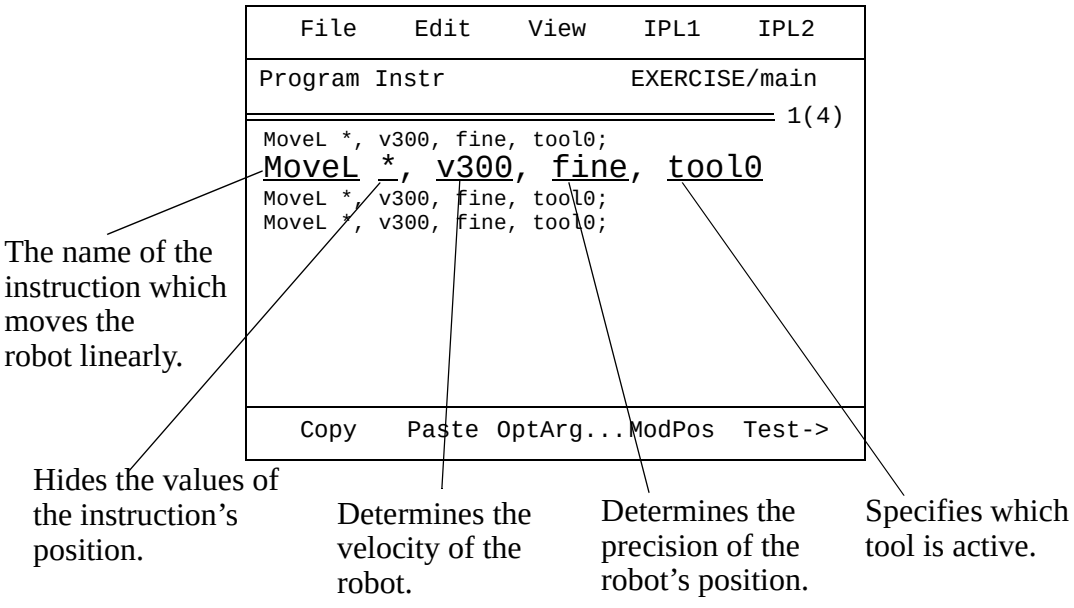

File Edit View IPL1 IPL2
Program Instr EXERCISE/main
1(4)
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
Copy Paste OptArg... ModPos Test->
The name of the instruction which moves the robot linearly.
Hides the values of the instruction’s position.
Determines the velocity of the robot.
Determines the precision of the robot’s position.
Specifies which tool is active.

Figure 33 Example of a motion instruction.

8 Starting the Program

You are now going to start the program you just opened. It should first be run step by step using reduced velocity, then continuously.

The program consists of four motion instructions and includes positions near the robot's "calibration position" (see Figure 34).

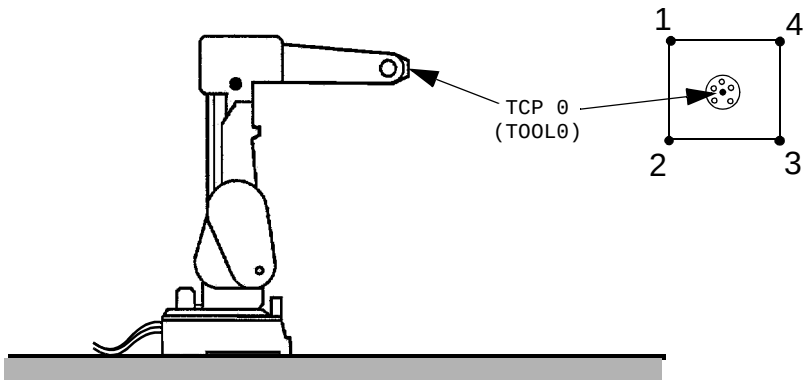


Figure 34 The robot's calibration position

Before starting the program move axis 5, manually with the joystick, downwards about 45°. (For information on the various robot axes, see Chapter 3.2)

1. Press the **Program** window key and a window, like the one in Figure 35, will appear (you have already reached this stage if you have come directly from Chapter 7).

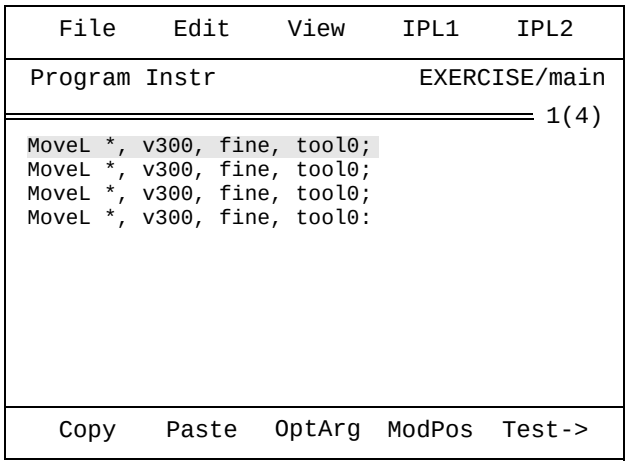

File Edit View IPL1 IPL2
Program Instr EXERCISE/main
1(4)
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0:
Copy Paste OptArg ModPos Test->

Figure 35 The Program window.

2. Press the ***Test*** function key. The window in Figure 36 appears.

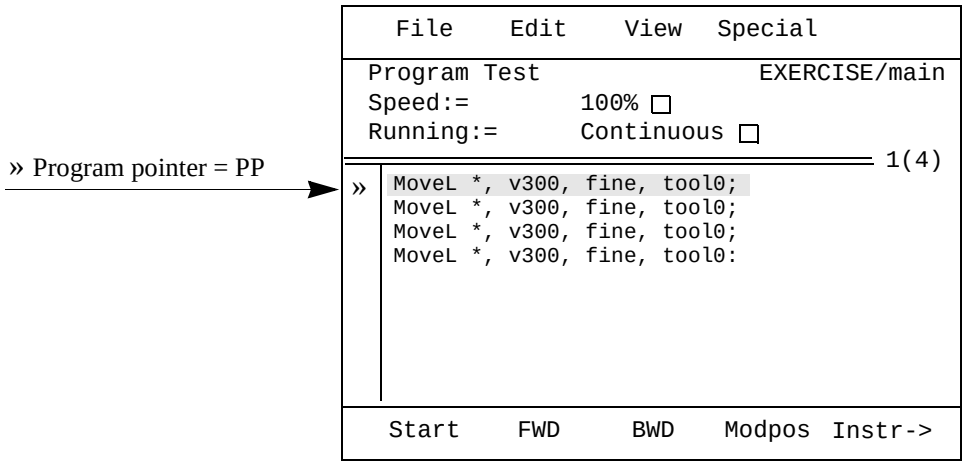
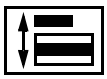


Figure 36 The Program Test window.

Function keys displayed:

- **Start**: continuous running of the program.
- **FWD**: one instruction forward.
- **BWD**: one instruction backward.
- **Instr->**: select the Program instruction window again.

The program point (PP) indicates the instruction with which the program will start when you press one of the options ***Start***, ***FWD*** or ***BWD***.

3. Select the upper part of the window by pressing the List  key**.**

4. Reduce the velocity to 75% by pressing the ***-%*** function key (see Figure 37). Correction is carried out in increments of 5%.

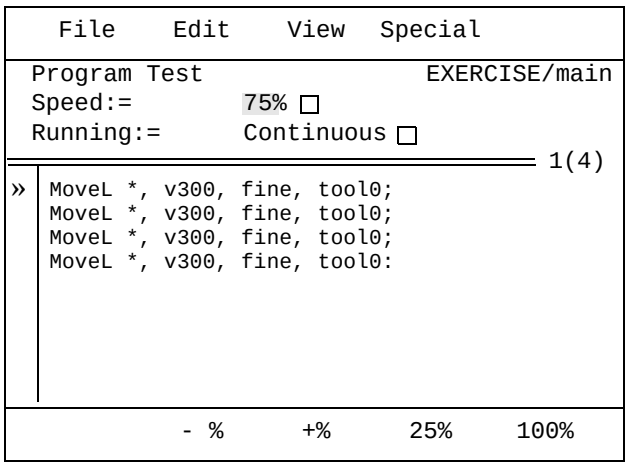


Figure 37 Correction of velocity.

5. Move (using the same key as in point 3) the cursor back to the first line of the program (see Figure 38).

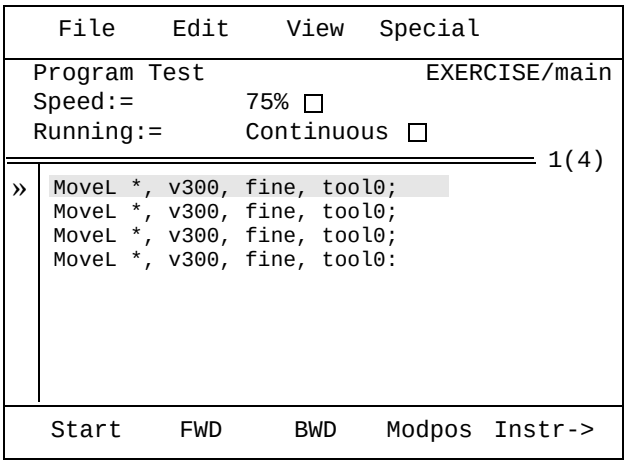


Figure 38 Window for starting the program.

The program can now be started. Make sure that no-one is inside the safeguarded space around the robot.

6. Start the program by pushing in the enabling device and pressing the ***FWD*** function key (see Figure 38).

   When the program has started, the robot will carry out one instruction, then it will stop. Press ***FWD*** to initiate the next instruction, press again for the next one, and so on.

   The window in Figure 39 is displayed during the execution of the program.

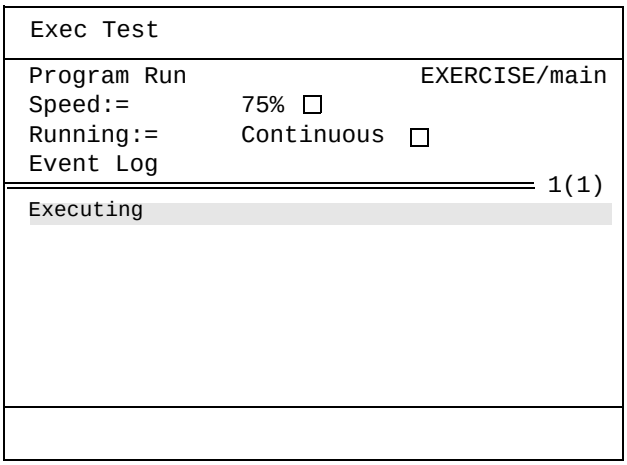


Figure 39 Window during program execution.

7. Go through all the program instructions step by step. Press ***FWD*** repeatedly after the robot is in position.

8. If you press ***FWD*** when the program comes to the final instruction, the program will start from the beginning again.

9. Let the robot move to position number 4 (see Figure 34).

10. Move, in the same way as before, the cursor to the **Running** field and change to **Cycle** execution.

11. Move the cursor back to the program.

12. Start the program by pressing ***Start***.

    When **Cycle** is selected the program will be executed once, and then will stop in position 4 (one cycle).

13. Select **Continuous** execution again.

9 Stopping the Program

Stop the program by pressing the **Stop** key on the teach pendant (see Figure 40).

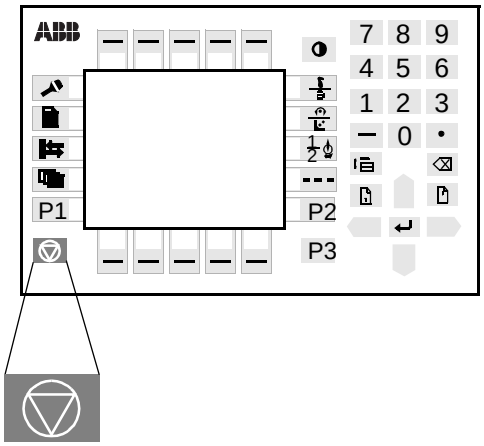


Figure 40 Stop key on the teach pendant.

10 Automatic Mode

Automatic mode is used to execute ready-made programs.

1. Turn the operating mode selector on the operator's panel to position . The window in Figure 41 appears.

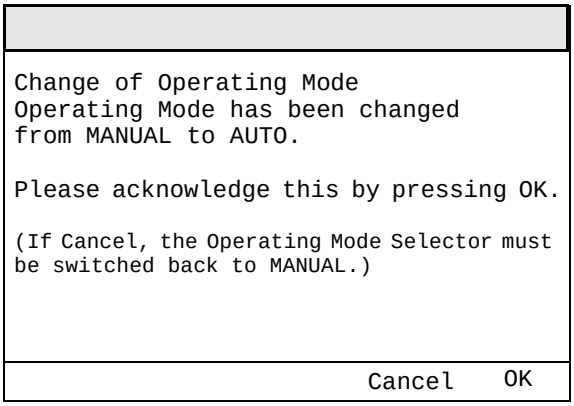

Change of Operating Mode
Operating Mode has been changed
from MANUAL to AUTO.
Please acknowledge this by pressing OK.
(If Cancel, the Operating Mode Selector must
be switched back to MANUAL.)
Cancel
OK

Figure 41 Dialog box used to confirm a change from manual to automatic mode.

2. Press ***OK***. You have now changed to automatic mode and the **Production** window appears on the display (see Figure 42).

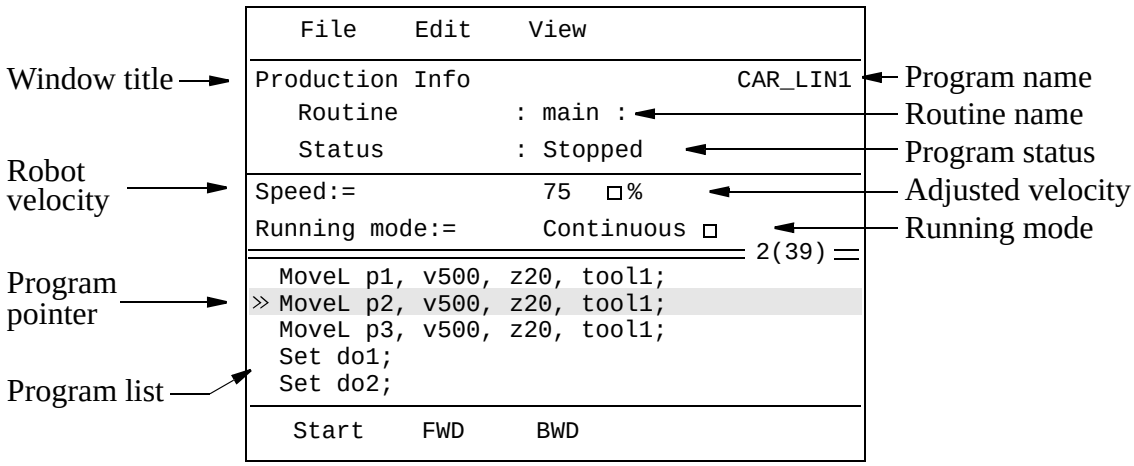


Figure 42 The Production window in auto mode.

3. Press the **MOTORS ON** button on the operator's panel.
4. Start the program with the ***Start*** function key.
5. Stop the program with the **STOP** button on the teach pendant.
6. Switch back to < 250 mm/s.

For further information, see Chapter 11, Production Running in the User's Guide.

11 Errors

A window displaying an error message appears whenever there is any type of error (see Figure 43).

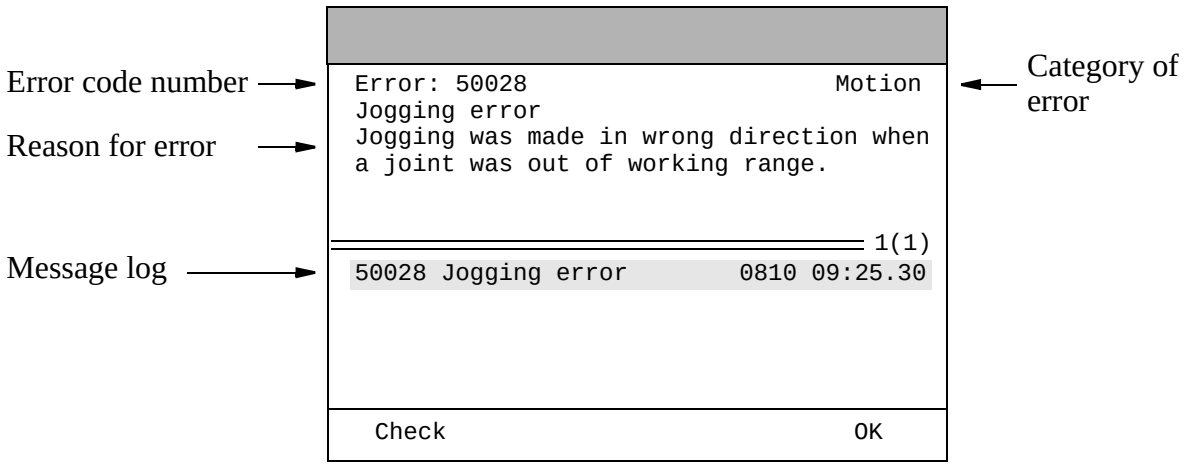


Figure 43 Example of a window displaying an error message.

Error code number

A number unique to each error.

Category of error

Assigns errors to groups relating to the type of error. Each category has its own code number series; e.g. Operator errors (6001-6999).

Reason

Describes the reason for the error in plain language. For more information regarding hardware faults, see the User's Guide - *16 Error Management*.

Message log

Indicates the most recent errors. The error indicated on the first line is displayed in the window. The log shows the error code number, a brief description of the error, and the time the error was registered. If you highlight one of the errors in the log, the window will then be updated with the appropriate error code number, reason and category.

The ***Check*** function key can be used to get help on how handle a specific error.

If you press ***OK***, the error-message window will disappear.

Using the joystick, try to manually operate the robot outside its operating area. You will then see an example of an error message.

12 Switching the robot off

If you are going to continue with the rest of the exercises, you can skip this chapter.

All output signals will be set to zero when the robot is switched off. This may affect the gripper and peripheral equipment.
So, before switching the robot off, check first that the equipment, and any people in the area, will not come to any harm.

1. If the program is running, stop it by pressing the **Stop** push button on the teach pendant.

2. After you have done this, switch off the mains switch.

   The robot's memory is battery-backed and is thus not affected when the system is switched off.

13 Changing a Program

The following chapters are intended to be read by people who will create programs, edit programs, etc.

This chapter explains some of the ways in which you can change the program you opened and started in the preceding chapters. You will:

- run the program step by step until you get to the position you want to modify
- modify this position
- change an argument in an instruction
- enter a new instruction (position)
- program a time delay (WaitTime)

13.1 Modifying positions

1. If you have exited the previous exercises, choose the **Program** window (see Figure 44).

Figure 44 The Program window key.

The window in Figure 45 appears on the display.

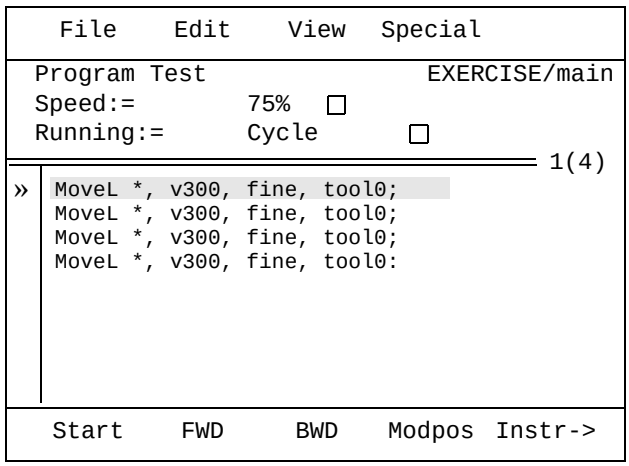


Figure 45 The Program Test window.

2. Push in the enabling device and press ***FWD***. Move the robot to the first position in the program (the first instruction should be highlighted).

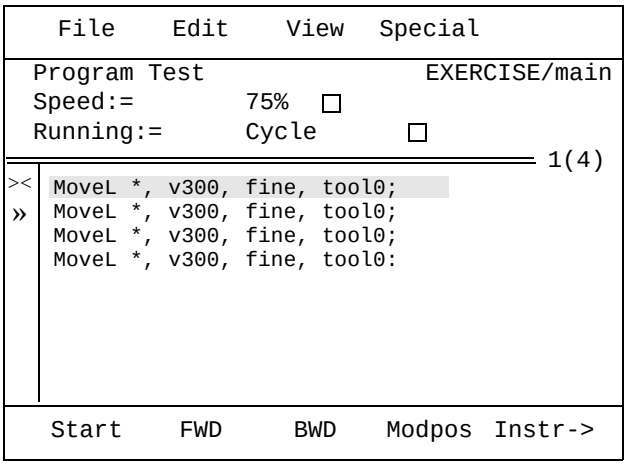


Figure 46 The first instruction is selected.

3. Then move the robot to a new position with the joystick.

4. Press the ***ModPos*** function key.

The window in Figure 47 will then appear on the dispaly.

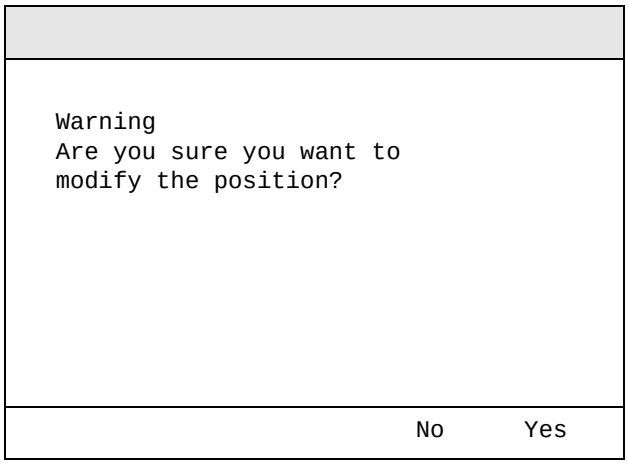


Figure 47 The warning dialog when modifying a position.

5. Press ***Yes*** if you want to change the original position specified to the current position of the robot.

6. Activate the enabling device and press ***FWD*** again to move the robot to the next position.

   Repeat points 3 to 6 and go through all the positions in the training program.

7. Test run the program step by step.
   Stop the program in any position and press the ***Instr*** function key (to terminate the Program Test). The window in Figure 48 will then appear on the display.

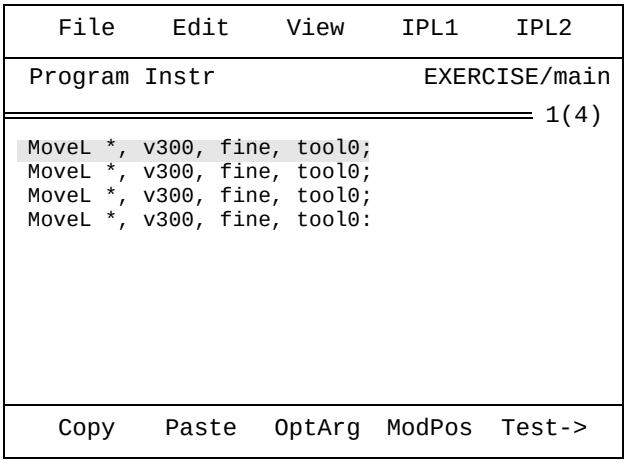

File Edit View IPL1 IPL2
Program Instr EXERCISE/main
1(4)
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0:
Copy Paste OptArg ModPos Test->

Figure 48 The Program Instr window.

13.2 Changing arguments

You are now going to change one of the arguments of the first move instruction (MoveL), which should be highlighted. You are going to change the precision of the position.

1. Select the "**fine**" argument (see Figure 49). Move the cursor using the right arrow key.

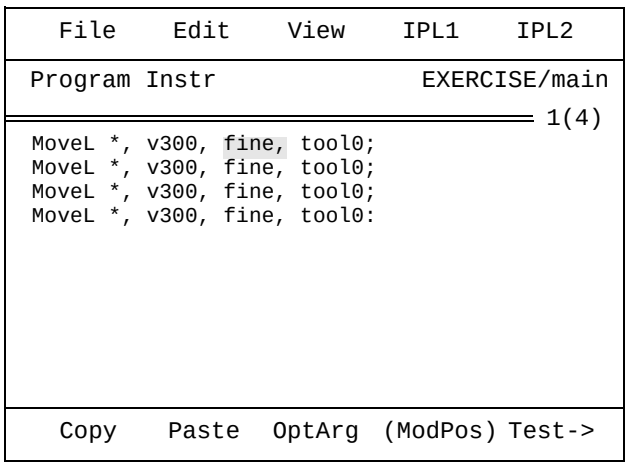

File Edit View IPL1 IPL2
Program Instr EXERCISE/main
1(4)
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0:
Copy Paste OptArg (ModPos) Test->

Figure 49 The "fine" argument is selected.

2. Press Enter .

The window in Figure 50 appears.

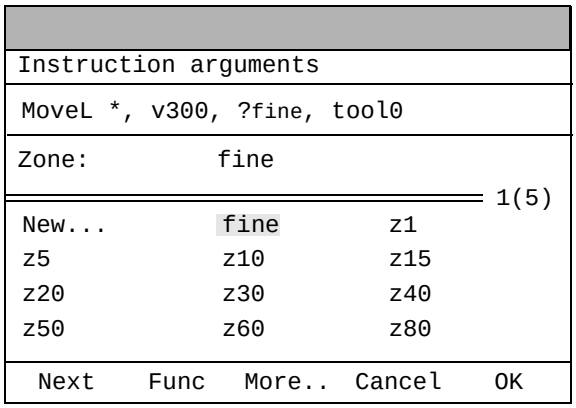


Figure 50 Dialog box for programming instruction arguments.

3. Move the cursor to **z10**.
4. Press Enter ↵ and the fine argument will change to z10.
5. Then press ***OK***. The instruction has now changed to z10.
6. Move the cursor so as to select the complete instruction (see Figure 51).

13.3 Adding instructions

You are now going to add a movement instruction to the program after the first instruction. The **Program Instr** window in Figure 51 should now appear on the display.

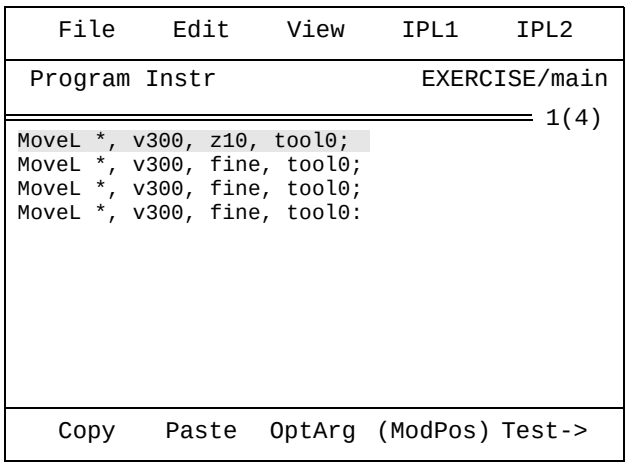

File Edit View IPL1 IPL2
Program Instr EXERCISE/main
1(4)
MoveL *, v300, z10, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0:
Copy Paste OptArg (ModPos) Test->

Figure 51 Program window.

1. Press the ***Copy*** function key to copy the first instruction (highlighted) in Figure 51.
2. Then press ***Paste***. The window in Figure 52 appears. As it is the first instruction in

the program that is highlighted, you will be asked where you want the new instruction to be inserted.

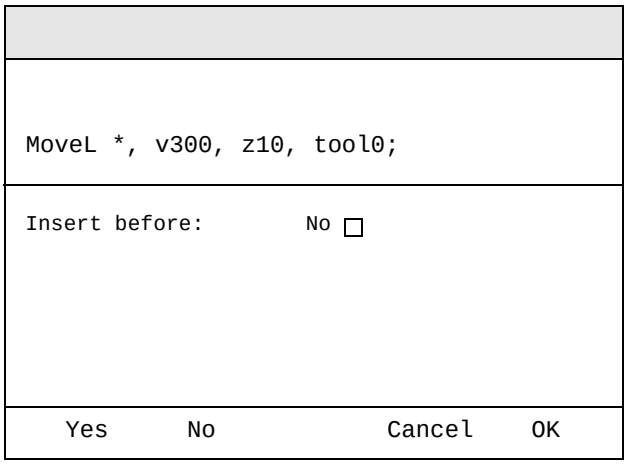


Figure 52 Dialog box used to insert new instructions when the first instruction is highlighted.

3. Select ***No***. Press ***OK***.
4. The new instruction will be inserted directly **under** the instruction that was highlighted, and will be highlighted itself.

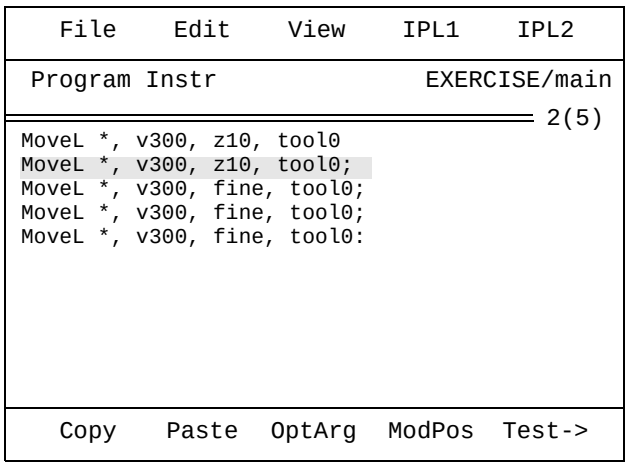

File Edit View IPL1 IPL2
Program Instr EXERCISE/main
2(5)
MoveL *, v300, z10, tool0
MoveL *, v300, z10, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0:
Copy Paste OptArg ModPos Test->

Figure 53 An extra position (the same as the one copied) is added to the program.

5. Using the joystick, move the robot to the position to which you want it moved.
6. Press ***ModPos*** (see Figure 53).
7. Test run the program using continuous execution.
8. Select ***Test->***.
9. Push the enabling device.
10. Press ***Start***.

13.4 Programming a delay

You are now going to program a delay, i.e. make the robot wait a specified amount of time. The new instruction will be inserted after the fourth instruction.

1. When the program test-run is completed, press the ***Instr*** function key.
   The window in Figure 54 appears.

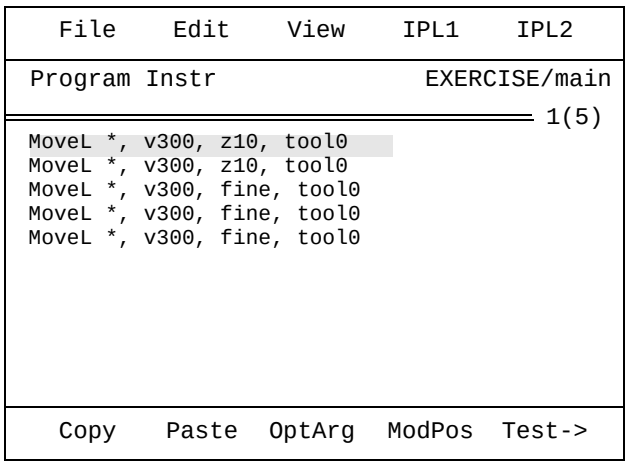


Figure 54 The Program Instruction window.

2. Using the arrow keys (up and down), move the cursor to the fourth instruction in the program. The new instruction will be inserted under the highlighted one.

3. Select **IPL1: Various**. The window in Figure 55 appears.

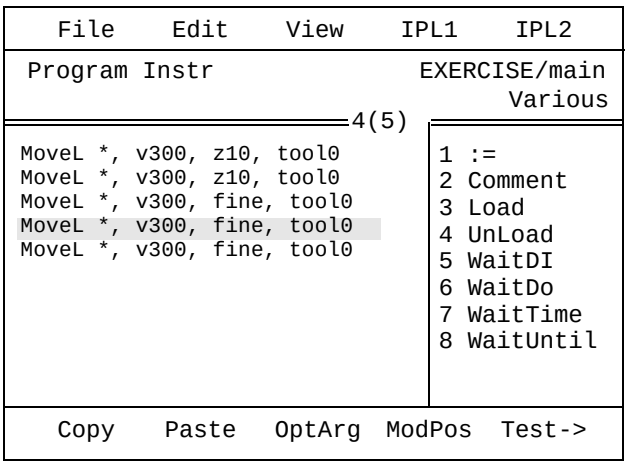
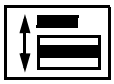


Figure 55 The pick list including the waitTime instruction.

4. Select the desired instruction from the pick list, in one of the following ways:

   - Using the numeric keyboard, enter the number (7) shown in front of WaitTime (see Figure 55). (The numeric keyboard is illustrated in chapter 3.)

- Select the pick list by pressing the **List** key  . Then select the desired instruction and press Enter .

5. A window like the one in Figure 56 appears.

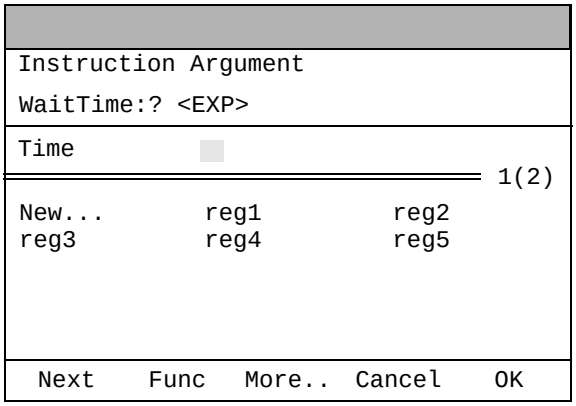


Figure 56 Dialog box for entering arguments.

6. Type 3 on the numeric keyboard to get a wait time of 3 seconds.
7. Press ***OK***. The window like the one in Figure 57 appears.

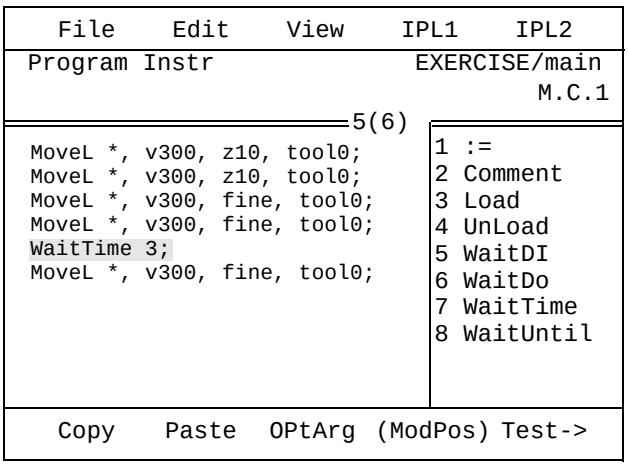


Figure 57 The Program Instr appears on the display.

8. Press **Edit: Hide IPL** to remove the pick list.
9. Now test run the program again using the Program Test window. Choose continuous execution.

INFORMATION

The **Edit** menu includes a number of functions which can be used to edit the program (see Figure 58).

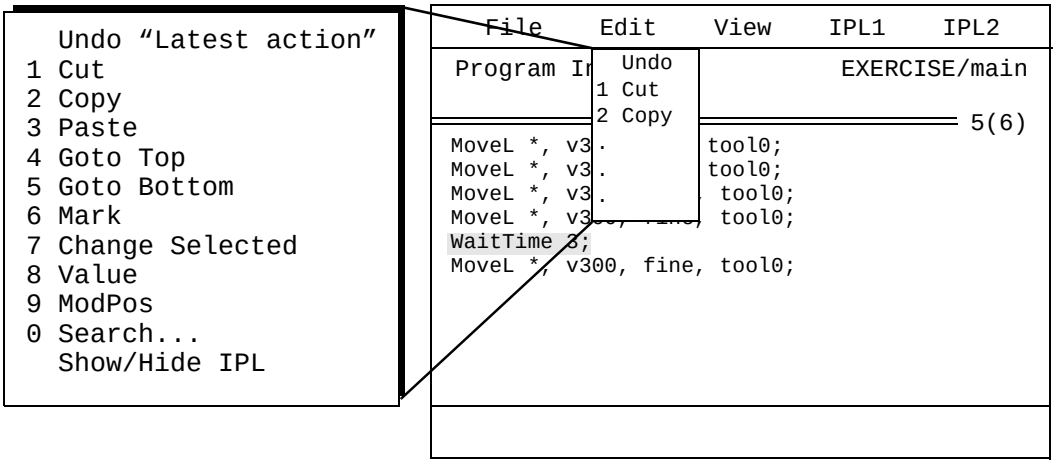


Figure 58 The Edit pull-down menu.

Press the **Edit** menu key again to remove the menu.

See the User’s Guide for more detailed information.

14 Storing the Program on Diskette

You are now going to copy the program to a diskette.
Use 3.5” HD (High Density) DOS formatted diskettes.

Do not use the Setup diskette to store the exercise program.

14.1 Storing on diskette

1. Choose the **Program Instr** application (see Figure 59), if you are not already in it.

Figure 59 The Program window appears on the display.

2. Press the **File** menu key. The window in Figure 60 appears.

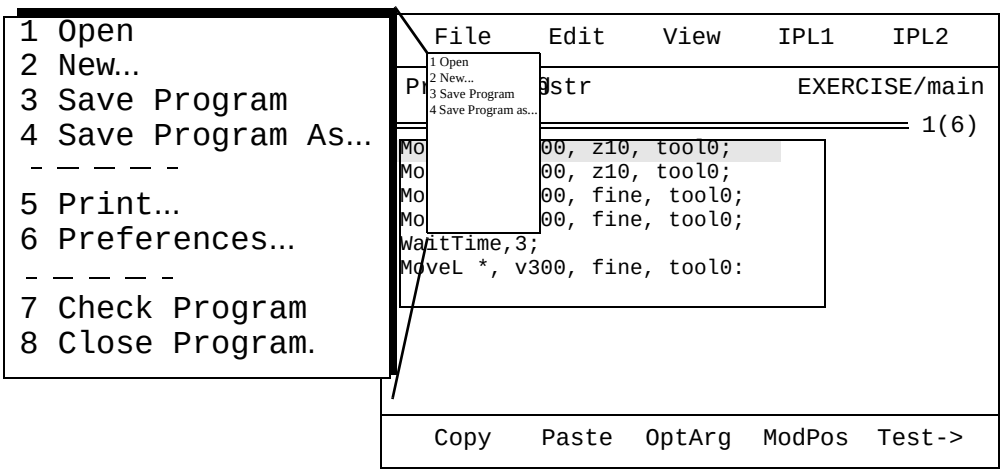


Figure 60 Commands in the File menu.

3. Select **File: Save Program as** and press Enter . (You could, alternatively, use the numeric keyboard to enter the number shown in front of the function name.) The dialog box in Figure 61 appears on the display.

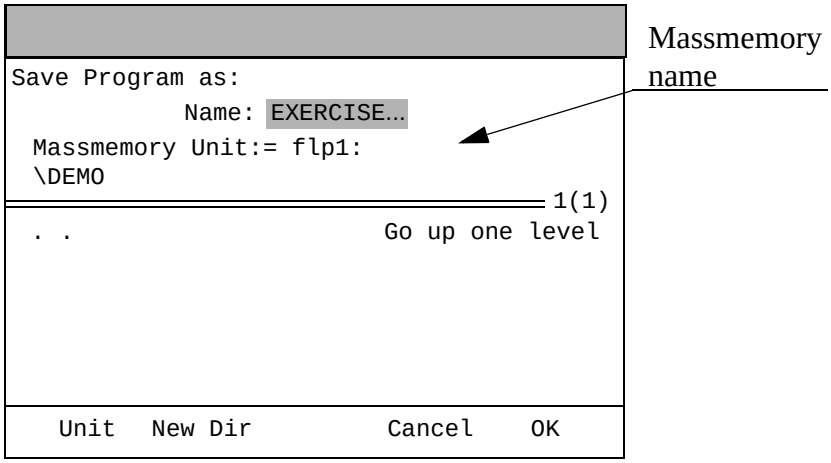


Figure 61 Dialog box for Save Program as.

4. Press ***Unit*** to choose the type of mass storage, if it is not already chosen; "flp1" should appear on the third line of the window (see Figure 61).

5. Press Enter ↵ to enter an optional name. The text-input dialog box in Figure 62 then appears.

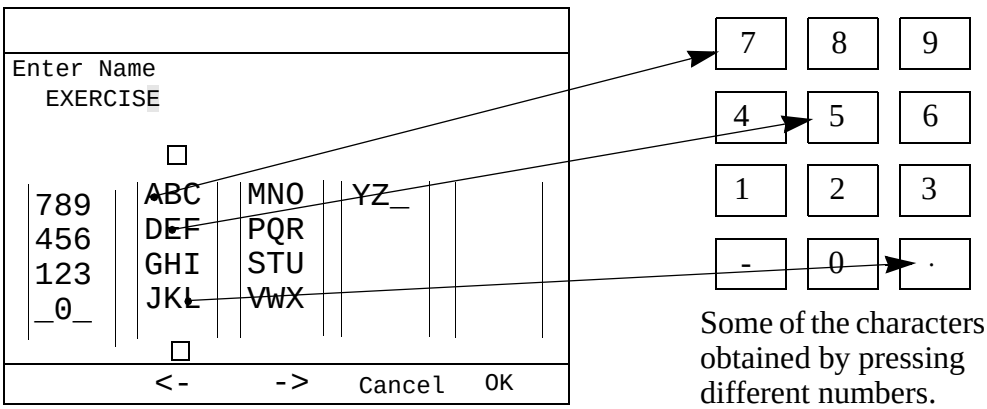


Figure 62 Window for entering text.

You can now see five groups of characters. Each group is represented on the numeric keyboard: the layout of the keyboard corresponds to the layout of the characters. You can move between the various groups using the <- and -> function keys (the selected group is indicated with squares both above and below it, see Figure 62).

Use the **Delete** ⌫ key to delete the name that is displayed or any errors you may type.

6. Now give the program a new name. When you have entered this, press ***OK.***
The window in Figure 63 appears.

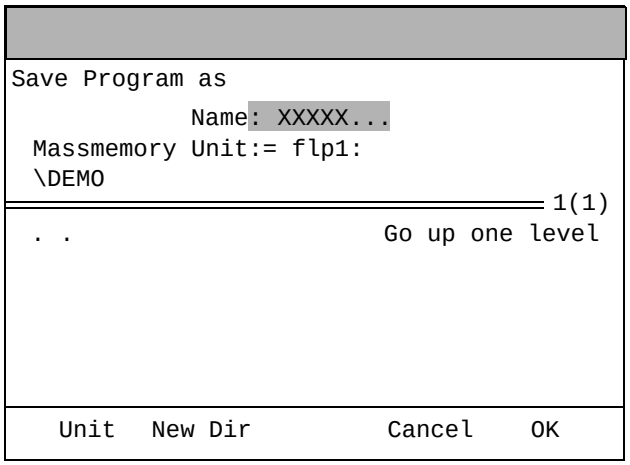


Figure 63 The Save Program as window.

7. Press ***OK***. A window like the one in Figure 64 appears.

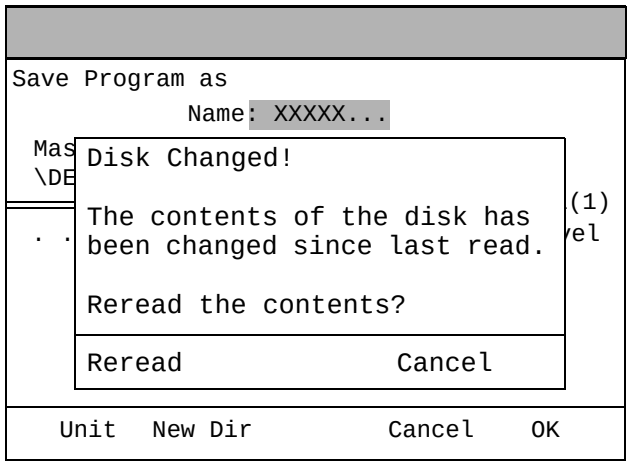

Save Program as
Name: XXXXX...
Disk Changed!
The contents of the disk has been changed since last read.
Reread the contents?
Reread Cancel
Unit New Dir Cancel OK

Figure 64 An alert box.

8. Press Enter ↵ . The window in Figure 65 appears.

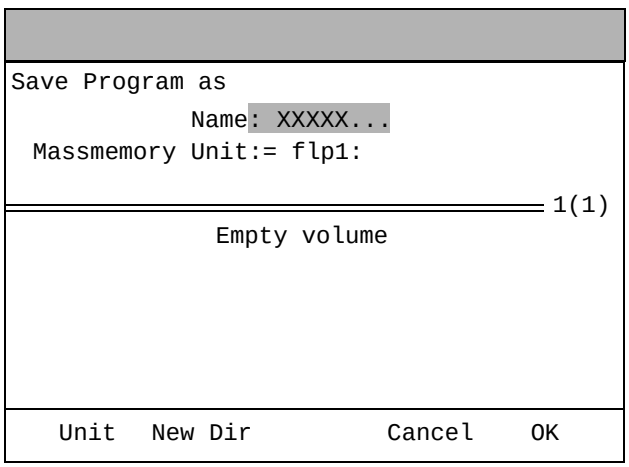


Figure 65 Window for storing the program on diskette.

9. The program is saved onto diskette when you press ***OK*** (see Figure 65).
10. Close the dialog box.

15 Printing Programs

15.1 Using a PC

It is also possible to print programs from a PC. Almost all word-processing programs can be used. The only requirement is that the computer can handle DOS-formatted diskettes.

1. Store the program on diskette.
2. Enter the program into the PC.
3. Print.

16 I/O Signals

This chapter describes how you can program an instruction which activates a digital output signal. After you have test run the program, you will manually open the I/O list and look at the signal in question.

16.1 Programming an I/O instruction

1. Select the **Program** window (see Figure 66).

Figure 66 The Program window key.

The window in Figure 67 appears on the display.

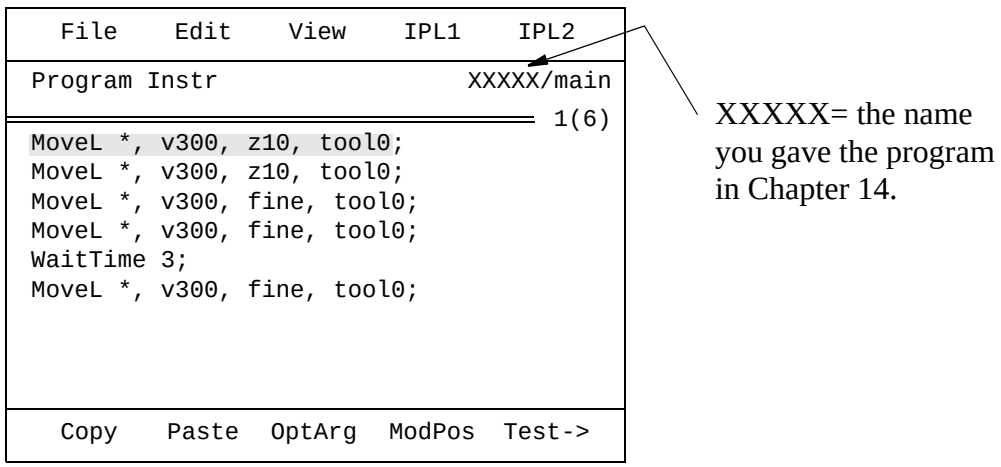

File Edit View IPL1 IPL2
Program Instr XXXXX/main
1(6)
MoveL *, v300, z10, tool0;
MoveL *, v300, z10, tool0;
MoveL *, v300, fine, tool0;
MoveL *, v300, fine, tool0;
WaitTime 3;
MoveL *, v300, fine, tool0;
Copy Paste OptArg ModPos Test->
XXXXX= the name you gave the program in Chapter 14.

Figure 67 The Program Instruction window.

The new instruction (set an output) will be entered directly **under** the highlighted instruction. Select the third instruction in the program.

2. Select **IPL1: IO**. The window in Figure 68 appears.

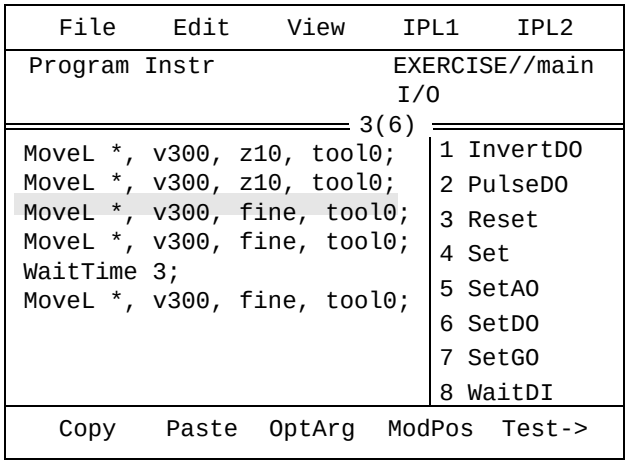


Figure 68 The pick list including the set instruction.

Select the function **Set** in the same way as you selected the WaitTime instruction in Chapter 13.

3. After you have selected the function **Set**, the dialog box in Figure 69 appears.

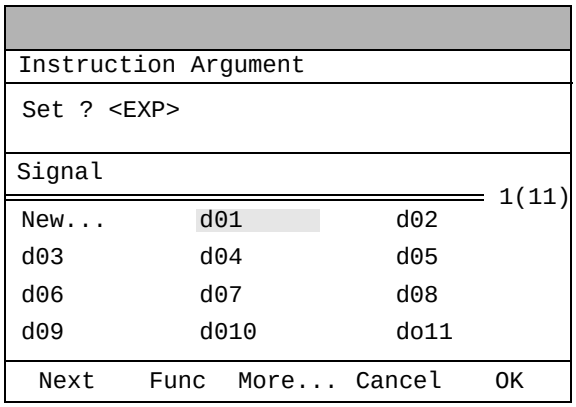


Figure 69 Dialog box for selecting a digital output signal.

4. Select “do4” in the list. Press ***OK***. The window in Figure 70 appears.

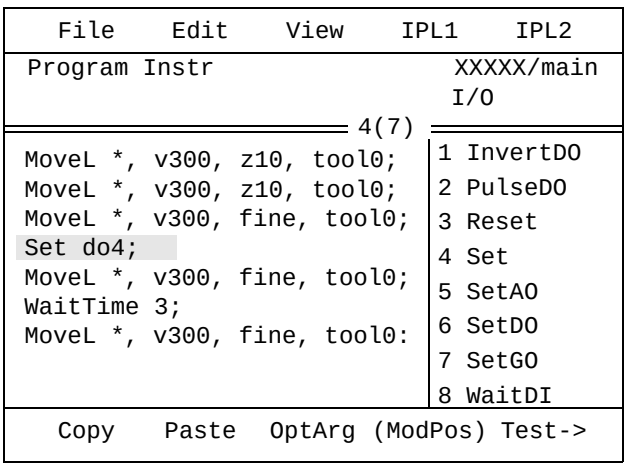


Figure 70 The “Set do4” instruction has been entered into the program.

5. Remove the pick list of instructions by pressing **Edit: Hide IPL**.
   The window in Figure 71 will then appear.

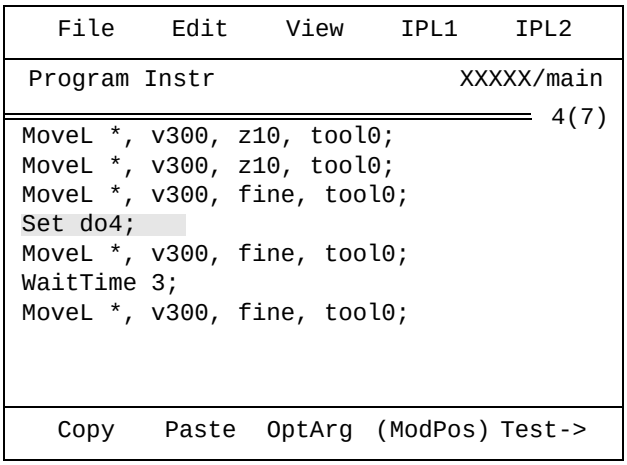
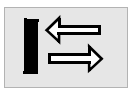


Figure 71 The Program Instruction window.

6. Test run the program using the ***Test*** function, one instruction at a time, so that the "Set do4" instruction can be read by the program.

7. You are now going to manually check the state of the signal.

8. Press the **Inputs/Outputs** window key (see Figure 72).



Inputs/Outputs window key

Previous page

Next page

Figure 72 The Manual I/O application.

   Browse through the list displayed in the window with the keys as shown in Figure 72.

9. Find (using the up and down arrow keys) the "do4" signal in the IO list and highlight it.

10. Check its value.

    You can change the value using the function keys (***0*** / ***1***) that appear on the display each time an output signal is highlighted.

11. Change the value of the signal and then press the Prog application key.

12. Test run the program once more (Test) and check the value of the signal again.

INFORMATION

When you use lists from the **View** menu in the Manual I/O window, you can choose to have only input signals, only output signals, etc., shown on the display.

CONTENTS

Page

1 Switching on the Power Supply 3
1.1 Errors on start-up 4
2 The Operator's Panel 4
3 Selecting the Operating Mode 4
3.1 Automatic mode (production mode) 4
3.2 Manual mode with reduced speed (programming mode) 5
3.3 Manual mode with full speed (testing mode) 5
4 Switching the Power Supply to the Motors On 5
5 Emergency Stops 6
5.1 Activating the emergency stop button 6
5.2 Resetting after an emergency stop 6
6 The Teach Pendant 7
6.1 Entering text using the teach pendant 9

Starting up

1 Switching on the Power Supply

Before switching on the power supply, check that no-one is in the safeguarded space around the robot.

• Switch on the mains switch

The robot hardware is then automatically checked. When the check is complete and if no errors have been detected, a message (see Figure 1) will be displayed on the teach pendant.

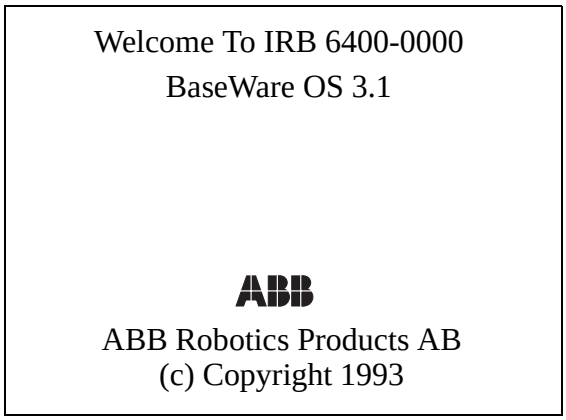


Figure 1 The welcome message after start-up.

In automatic mode, the Production window will appear after a few seconds.

The robot is started up with the same status as when the power was switched off. The program pointer remains unchanged and all digital outputs are set to the value before power off, or to the value specified in the system parameters. When the program is restarted, this is considered to be a normal stop - start:

- The robot moves back slowly to the programmed path (if there is a deviation) and then continues on the programmed path.
- Motion settings and data are automatically set to the same values as before power off.
- The robot will continue to react on interrupts.
- The mechanical units that was active before power off will automatically be activated at program start.
- The arc welding and spot welding processes are automatically restarted. But if a change of weld data has just been executed, this new data will be activated too early on the seam.

Limitations:

- All files and serial channels are closed (this can be handled by the user program).
- All analogue outputs are set to 0 and the Soft servo/Tune servo is set to default values (can be handled by the user program).
- WeldGuide cannot be restarted.

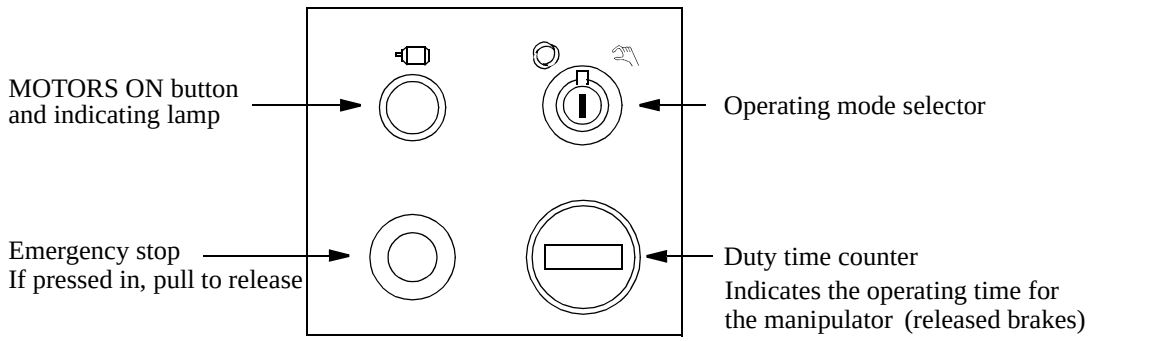
- Independent axes cannot be restarted.
- If the power failure occurs during a movement in an interrupt routine or error handler, the restart of the path is not possible.
- If the program execution is in a part when the CPU is very busy, there is a small chance that there is not enough time to make a proper close down at power failure. The robot will in this case tell the user that a restart is not possible.

1.1 Errors on start-up

During the entire start-up sequence, the robot functions are checked extensively. If an error occurs, it is reported as a message in plain text on the teach pendant, and recorded in the robot's event log. For more information on troubleshooting, see the Product Manual.

2 The Operator's Panel

The functions of the operator's panel are described in Figure 2.



MOTORS ON
Continuous light = Ready for program execution
Fast flashing light (4Hz) = The robot is not calibrated or the revolution counters are not updated
NB: The motors have been switched on
Slow flashing light (1 Hz) = One of the safeguarded space stops is active
NB: The motors have been switched off

Figure 2 The operator's panel is located on the front of the cabinet.

3 Selecting the Operating Mode

The operating mode is selected using the operating mode selector.

3.1 Automatic mode (production mode)

When the robot is in the automatic operating mode, it is essential that nobody enters the safeguarded space around it. Carelessness may cause personal injury.

• Turn the key to

Automatic mode is used when running complete programs in production operation. In this mode, the enabling device on the teach pendant is disconnected and the functions used to edit programs are locked.

3.2 Manual mode with reduced speed (programming mode)

• Turn the operating mode selector to

If the hold-to-run function is active (the function is available by means of a system parameter), program execution will stop as soon as you release the Hold-to-run key on the teach pendant.

Manual mode with reduced speed is used when programming and when working in the robot working space. In this mode, external units cannot be remotely controlled.

3.3 Manual mode with full speed (testing mode)

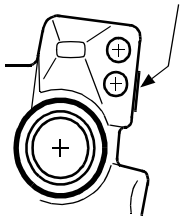
In Manual 100% mode, the robot moves at full speed. This operating mode may only be used by trained personnel. Carelessness may cause personal injury.

• Turn the operating mode selector to 100%

The hold-to-run function is now active, i.e. program execution will stop as soon as you release the Start key on the teach pendant.

Manual mode with full speed is only used when testing the robot program at full speed. In this mode, external units cannot be remotely controlled.

4 Switching the Power Supply to the Motors On



• In automatic mode, press the Motors On button on the operator's panel.

• In manual mode, turn to MOTORS ON mode by pressing the enabling device on the teach pendant halfway in.

If the enabling device is released and pressed again within half a second, the robot will not return to the MOTORS ON state. If this happens, first release the enabling device, then push it halfway in again.

5 Emergency Stops

5.1 Activating the emergency stop button

Emergency stop buttons are located on the operator's panel and on the teach pendant. There are often other ways of activating an emergency stop, but these depend on the robot installation.

When the emergency stop button is activated, the power supply to the motors is shut off and program execution stops.

5.2 Resetting after an emergency stop

- Fix the problem that caused the emergency stop.
- Reset the emergency stop state by pressing the MOTORS ON button (see Figure 3).

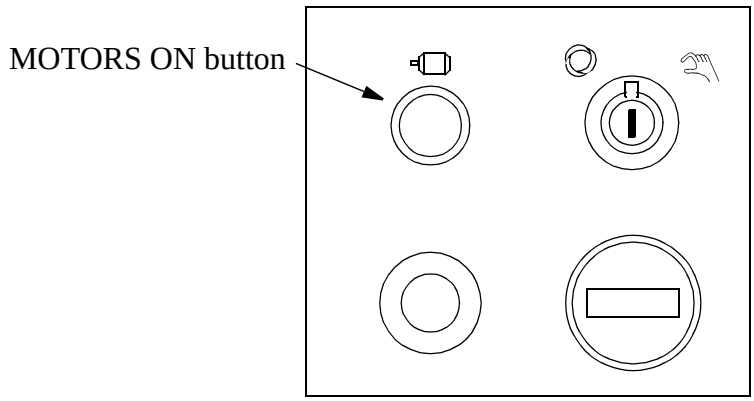


Figure 3 The emergency stop must be reset before setting the robot in the MOTOR ON state.

6 The Teach Pendant

The teach pendant is described below, see Figure 4.

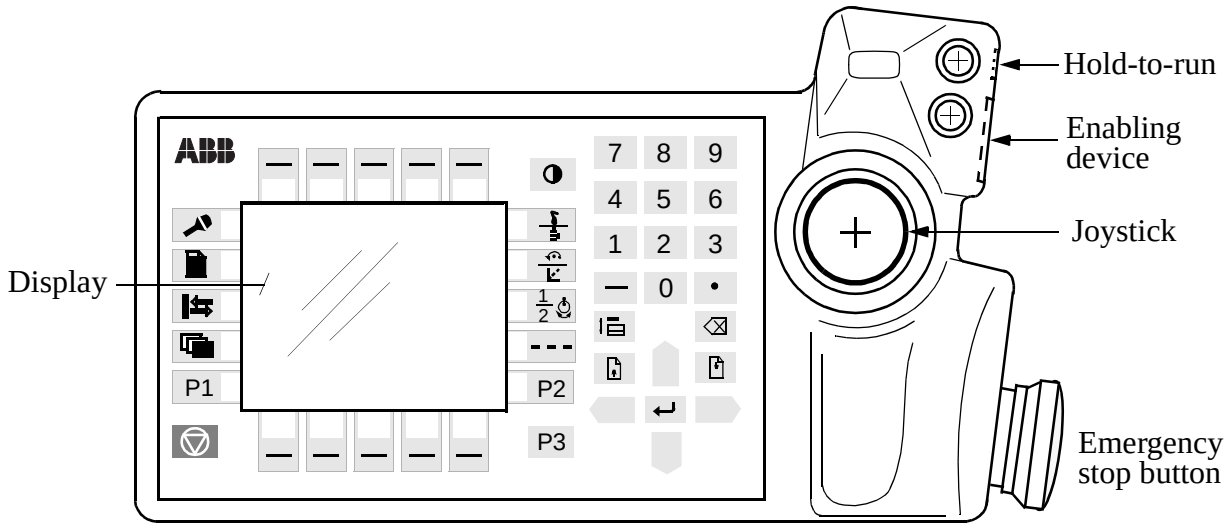
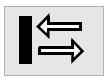


Figure 4 The teach pendant is used to operate the robot.

Jogging: Used to jog the robot.

Program: Used to program and test.



Inputs/Outputs: Used to manually operate the input and output signals connected to the robot.

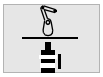
Misc.: Miscellaneous; other windows, i.e. the System Parameters, Service, Production and File Manager windows.

Stop: Stops program execution.

Contrast: Adjusts the contrast of the display.

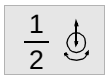
Menu keys: Press to display menus containing various commands.

Function keys: Press to select the various commands directly.



Motion Unit: Press to jog the robot or other mechanical units.

Motion Type: Press to select how the robot should be jogged, reorientation or linear.



Motion Type: Axis by axis movement. 1 = axis 1-3, 2 = axis 4-6

Incremental: Incremental jogging on/off

List: Press to move the cursor from one part of the window to another (normally separated by a double line).

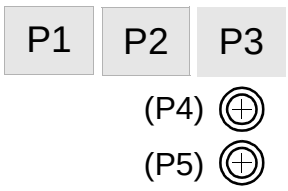
Previous/Next page: Press to see the next/previous page.

Delete: Deletes the data selected on the display.

Enter: Press to input data.

Up and Down arrows: Press to move the cursor up or down.

Left and Right arrows: Press to move the cursor to the left or right.



User defined keys: How to define these, see Chapter 12, *System Parameters*

6.1 Entering text using the teach pendant

When naming files, routines, data, etc., text can be entered using the teach pendant. As there is no character keyboard available, the numeric keyboard is used in a special way (see Figure 5).

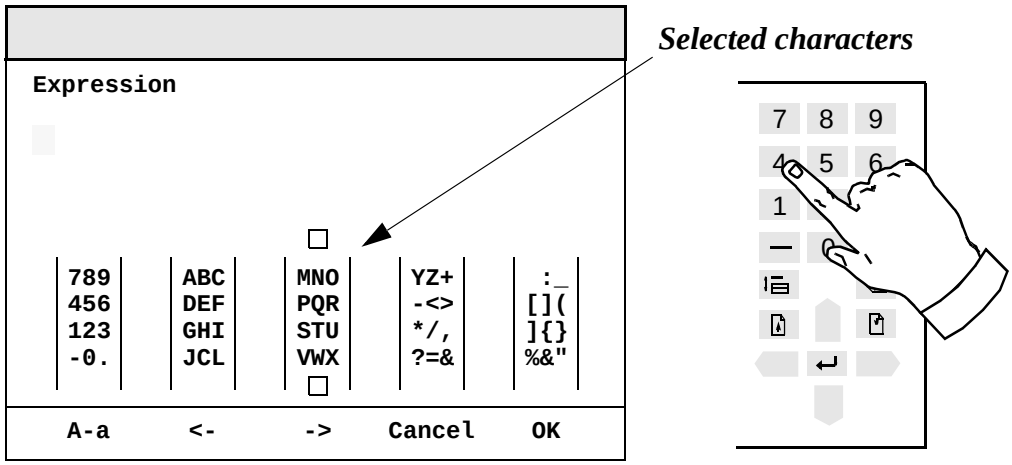


Figure 5 The dialog box used for entering text.

The keys on the numeric keyboard correspond to the selected characters on the display.

- Select a group of characters by pressing the function key -> or <-.
- Press the corresponding key on the numeric keyboard. If the third group is selected (as shown in Figure 5), 7 corresponds to M, 8 to N, 9 to O, etc.
- Move the cursor to the right or left using ArrowLeft ⟨ or ArrowRight ⟩ .
- Delete the character in front of the cursor by pressing Delete ⌫ .
- Switch between upper and lower case letters by pressing ***A-a***.
- When you have finished entering text, press ***OK***.

CONTENTS

Page

1 General 3
1.1 The Jogging window 3
1.2 Reading the current position 4
1.3 How moving the joystick affects movements 4
1.4 Locking of joystick axes 5
1.5 Motion Supervision 5
2 Jogging the Robot 6
2.1 Jogging the robot along one of the base coordinate axes 6
2.2 Jogging the robot in the direction of the tool 7
2.3 Reorienting the tool 9
2.4 Aligning a tool along a coordinate axis 9
2.5 Jogging the robot in the direction of the work object 11
2.6 Jogging the robot along one of the world coordinate axes 13
2.7 Using a stationary tool 13
2.8 Jogging the robot axis-by-axis 14
2.9 Incremental movement 14
2.10 Jogging an unsynchronised axis 15
3 Jogging External Axes 16
3.1 Choosing external units 16
3.2 Jogging external units axis-by-axis 16
3.3 Jogging external units coordinated 17

Jogging

Jogging

1 General

A joystick is used to jog the robot. It has three degrees of freedom, which means that you can move the robot in three different directions simultaneously. The robot speed is proportional to the joystick deflection, the greater the joystick deflection, the higher the speed (but not faster than 250 mm/s).

The joystick can be used irrespective of which window is open at the time. You cannot, however, jog the robot in the following situations:

- When the robot is in automatic mode .
- When the robot is in the MOTORS OFF state.
- When program execution is in progress.

If any axis is outside its working range, it can only be jogged back into its working range.

The function of the joystick can be read from and changed in the Jogging window. Some of the settings can also be changed directly using the motion keys on the teach pendant, see Figure 1.

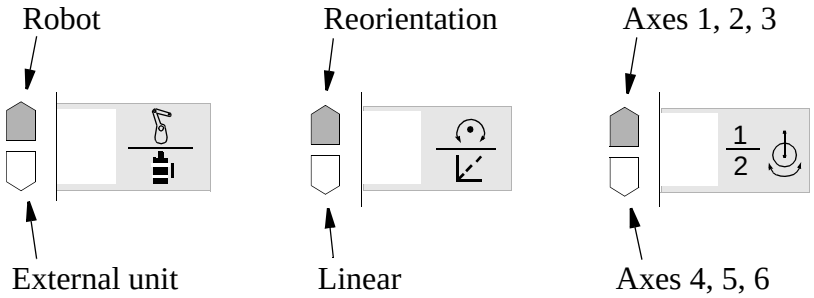


Figure 1 The indications next to the motion keys show the current settings.

The robot or external unit will start to move immediately you move the joystick. Make sure that no one is standing in the safeguarded space around the robot and also that the motion settings for jogging are correctly set. Carelessness can injure someone or damage the robot or other equipment.

1.1 The Jogging window

• Press the Jogging key to open the window.

The window displays the current motion settings for jogging and the current position of the robot. See the example in Figure 2.

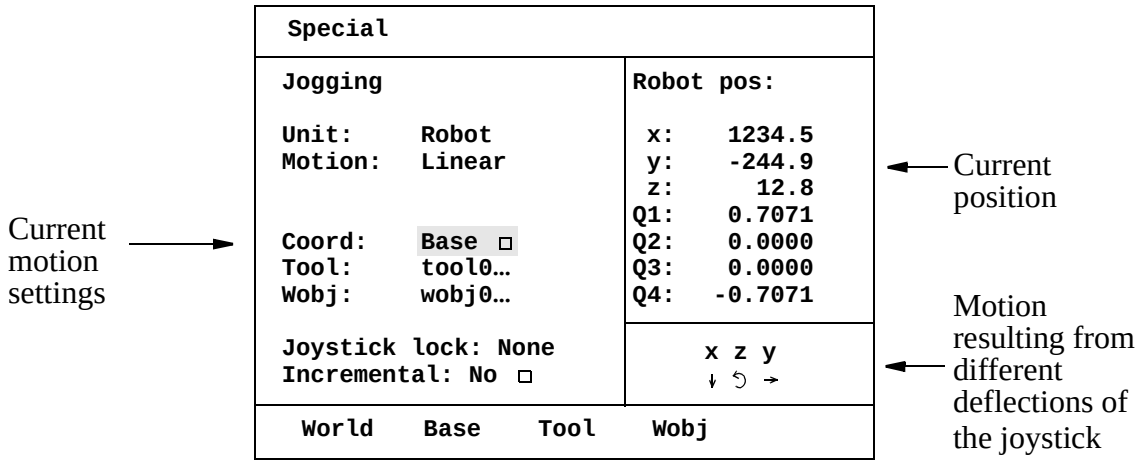


Figure 2 Define the various jogging settings in the Jogging window.

1.2 Reading the current position

The current position of the robot is displayed in the Jogging window (see Figure 2).

In *Linear* or *Reorientation* motion types, the position and orientation of the tool in relation to the coordinate system of the chosen work object is displayed (regardless of the type of coordinate system used).

In *Axis-by-Axis* motion type with *Robot* as the unit, the positions of the robot axes are displayed in degrees related to the calibration position of the respective axis.

When an external unit is moved, the position of the axes is displayed. In the case of linear axes, the position is displayed in mm related to the calibration position. For rotating axes, the position is displayed in degrees related to the calibration position.

When a unit is unsynchronised, no position is displayed.

1.3 How moving the joystick affects movements

The field that indicates the various deflections of the joystick displays how the principal joystick directions are linked to axes or coordinate directions. See the example in Figure 3.

Note The relationship between joystick deflection and robot movement can be changed in the system parameters. All the figures in this manual are related to standard setup.

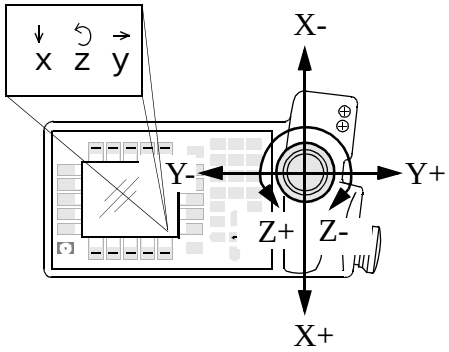


Figure 3 The direction of movements associated with each joystick deflection is displayed in the Jogging window.

1.4 Locking of joystick axes

It is possible to disable joystick deflections in certain directions.

- Select the field ***Joystick lock*** (see Figure 4).
- Select the joystick axes to be disabled by pressing the corresponding function key.

Enable all axes by pressing the function key ***None***.

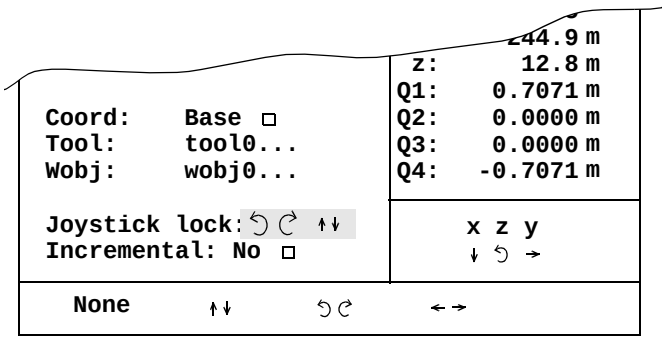


Figure 4 A joystick with disabled up-down and rotational deflection.

1.5 Motion Supervision

It is possible from the jogging window to turn motion supervision (collision detection) on and off. This will only affect motion supervision during jogging.

- Choose **Special: Motion Supervision.**

A dialog box appears displaying the setting for motion supervision (see Figure 5).

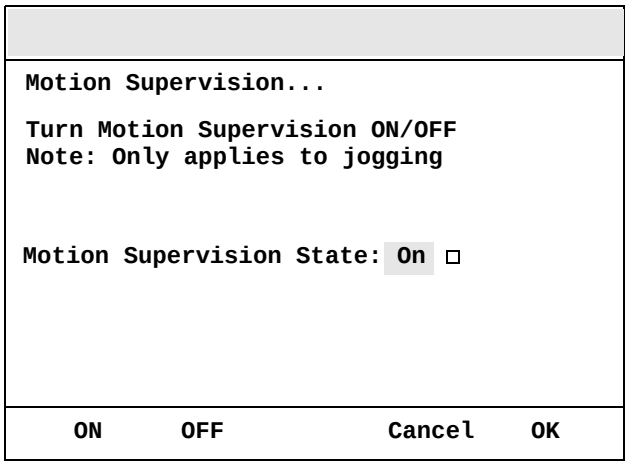


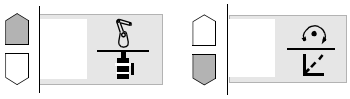
Figure 5 The dialog box for motion supervision

If you want to turn the motion supervision off or on:

- Press the function key ***OFF*** or ***ON***,
- Press ***OK*** to confirm.

2 Jogging the Robot

2.1 Jogging the robot along one of the base coordinate axes

- Set the keys  to jog the robot in a straight line.
- Select the field ***Coord*** (see Figure 6).
- Press the function key ***Base***.

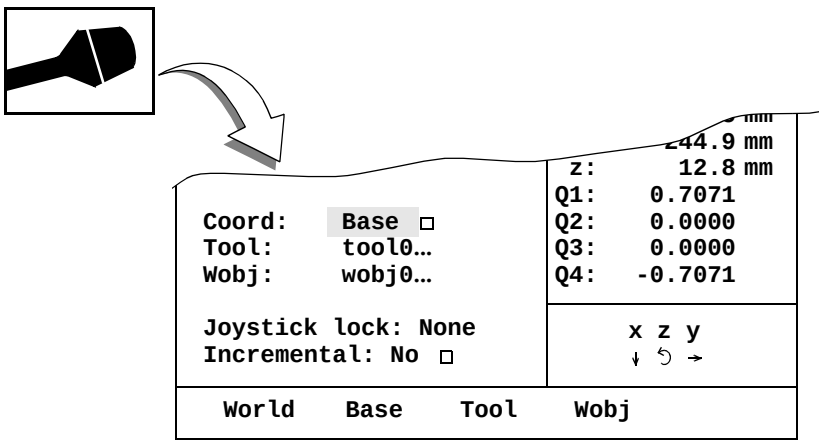


Figure 6 Specify the coordinate system in the Jogging window.

The robot will move the TCP along the base coordinate axes (see Figure 7).

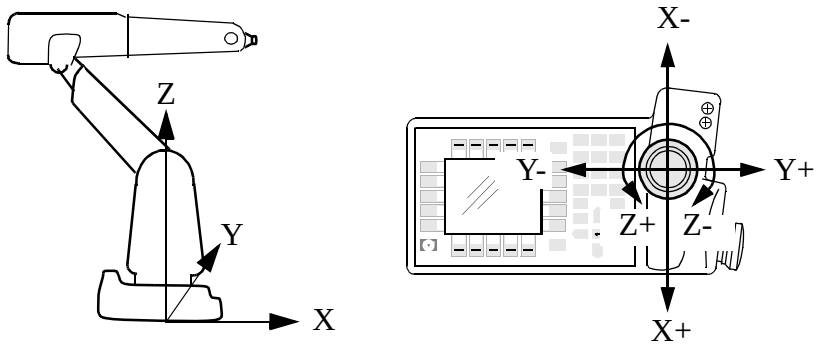
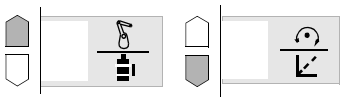


Figure 7 Linear movement in the base coordinate system.

2.2 Jogging the robot in the direction of the tool

• Set the keys  to jog the robot in a straight line.

• Select the field ***Coord*** (see Figure 8).

• Press the function key ***Tool***.

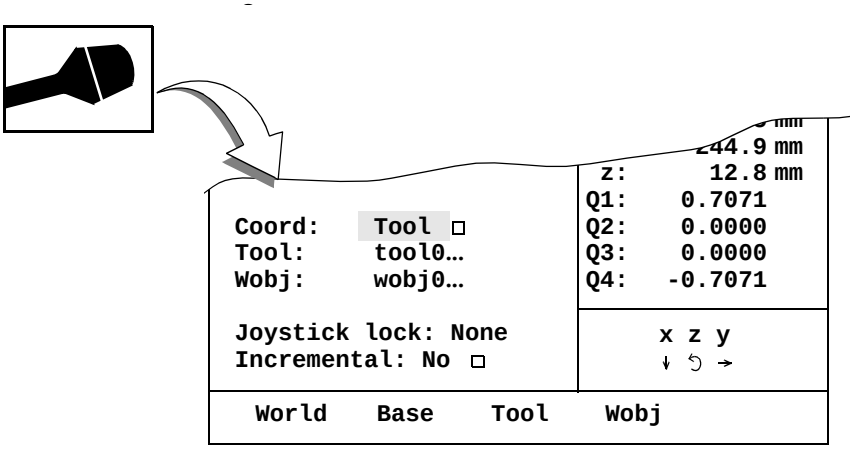


Figure 8 Specify the coordinate system in the Jogging window.

The tool that was last used when jogging the robot or last used for program execution is automatically chosen (see Figure 9).

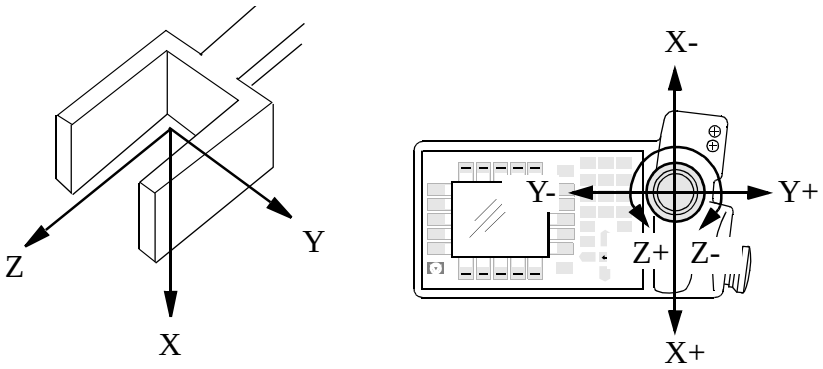


Figure 9 Linear movement in the tool coordinate system.

If you want to change the tool:

• Select the field ***Tool*** (see Figure 10).

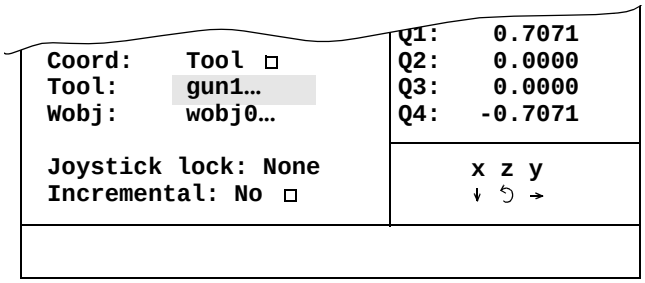


*Figure 10 Choose a tool by selecting the field **Tool**.*

• Press Enter ↵ .

• Select the desired tool from the dialog box which subsequently appears on the display. (*Tool0* in the dialog box corresponds to the centre of the mounting flange.)

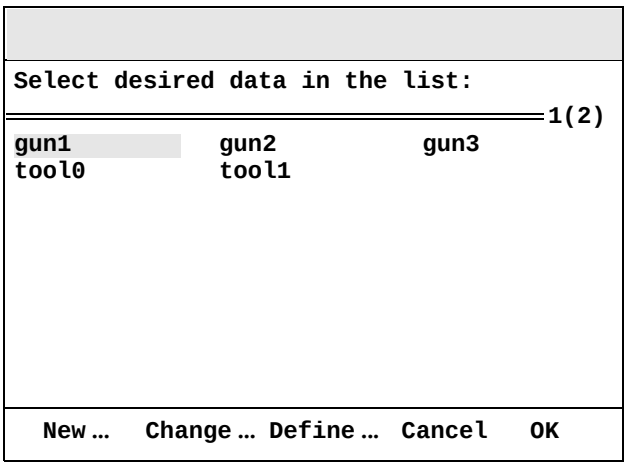


Figure 11 Changing or adding a tool.

You can create a new tool as follows:

• Press ***New.***

You can change the values of a tool as follows:

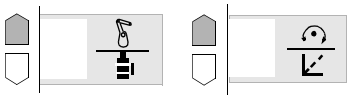
• Press

- ***Change*** to input the value manually
- ***Define*** to use the robot to measure up the tool coordinate system.

For more information see Chapter 10 Calibration.

• Press ***OK*** to confirm.

2.3 Reorienting the tool

• Set the keys  to reorientate the tool.

The tool is reorientated about the axes of the coordinate system that was chosen. The TCP of the chosen tool will not move (see Figure 12).

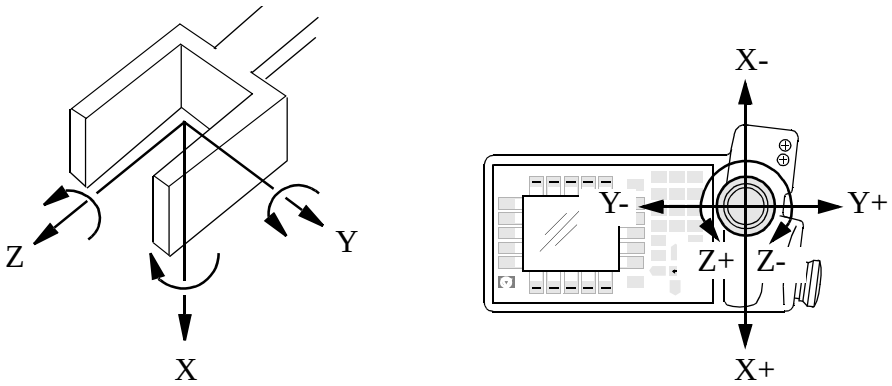


Figure 12 Reorientation about the tool coordinate system's axes.

2.4 Aligning a tool along a coordinate axis

The Z-direction of a tool can be aligned along a coordinate axis of a chosen coordinate system. The angle between the tool's Z-direction and the coordinate axes determines which coordinate axis the tool should be aligned along; the axis closest to the tool's Z-direction will be used (see Figure 13).

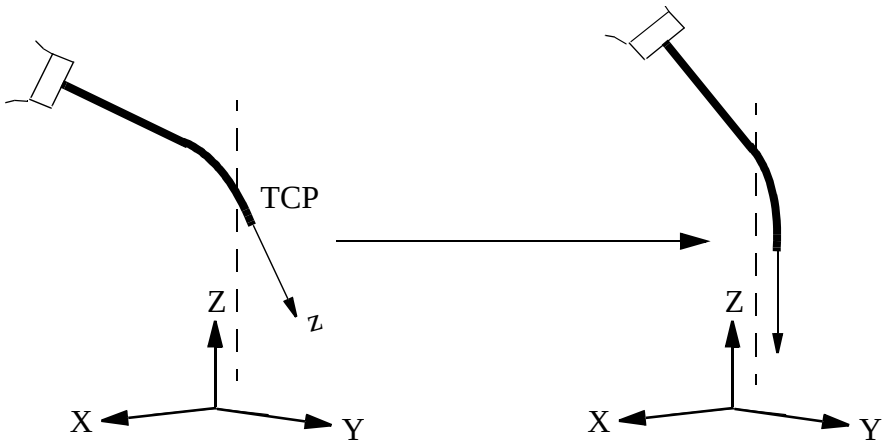


Figure 13 The tool is aligned along the Z-axis.

Adjust the direction of the tool manually so that it is close to the desired direction.

• Choose **Special: Align**

A dialog box appears displaying the coordinate system used for alignment (see Figure 14).

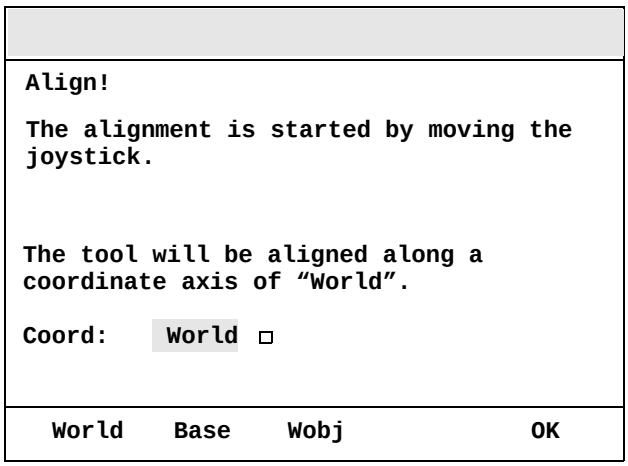


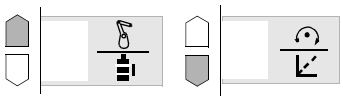
Figure 14 The dialog box for aligning the tool.

If you want to change the coordinate system, press any of the function keys ***World, Base*** or ***Wobj.***

• To start the alignment, press the enabling device and move the joystick. The joystick is used to adjust the speed. The robot will automatically stop as soon as it reaches the desired position.

• Press ***OK*** to confirm.

2.5 Jogging the robot in the direction of the work object

• Set the keys  to jog the robot in a straight line.

• Select the field ***Coord*** (see Figure 15).

• Press the function key ***Wobj***.

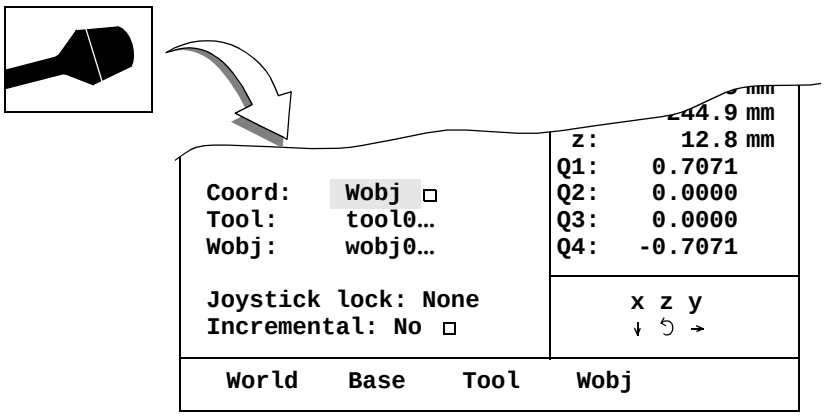


Figure 15 Specify the coordinate system in the Jogging window.

The work object that was last used when jogging the robot or last used for program execution is automatically chosen.

If you want to change the work object:

• Select the field ***Wobj*** (see Figure 16).

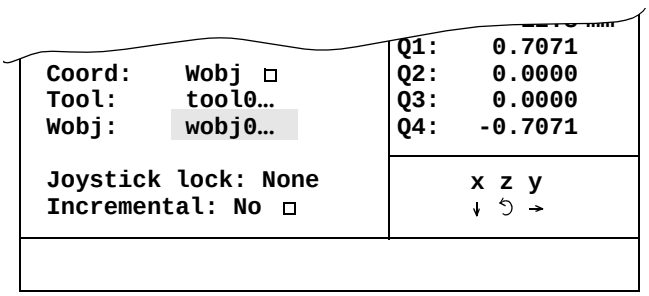


*Figure 16 Choose a work object by selecting the field **Wobj**.*

• Press Enter ↵.

• Select the desired work object from the dialog box which subsequently appears on the display. (*Wobj0* in the dialog box corresponds to the world coordinate system.)

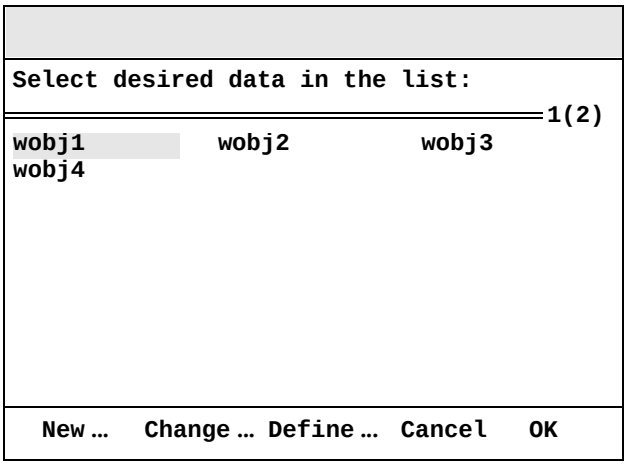


Figure 17 Changing or adding a work object.

You can create a new work object as follows:

• Press ***New.***

You can change the values of a work object as follows:

• Press

- ***Change*** to input the value manually
- ***Define*** to use the robot to measure up the coordinate systems.

For more information see chapter 10 Calibration.

• Press ***OK*** to confirm.

The robot will move along the axes of the object coordinate system (see Figure 18).

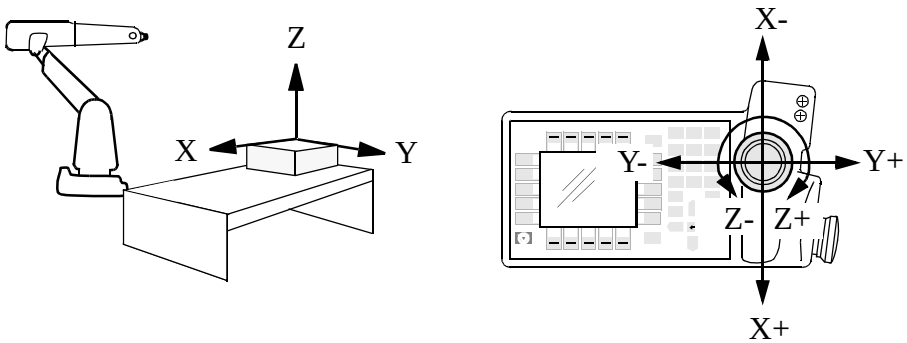
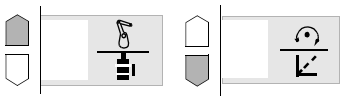


Figure 18 Linear movement in the object coordinate system.

2.6 Jogging the robot along one of the world coordinate axes

• Set the keys  to jog the robot in a straight line.

• Select the field ***Coord*** (see Figure 19).

• Press the function key ***World***.

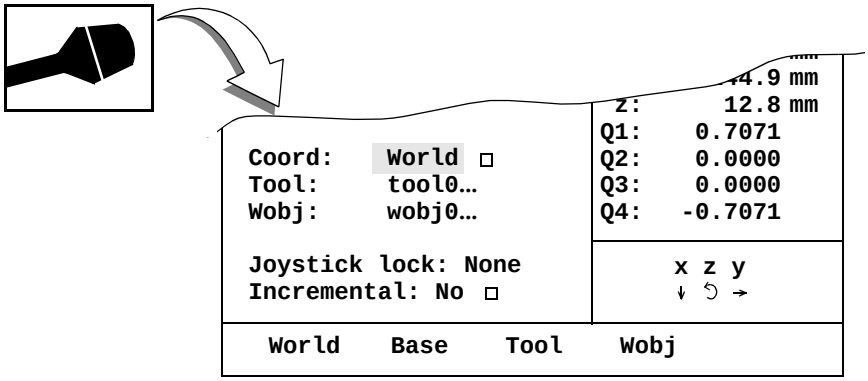


Figure 19 Specify the coordinate system in the Jogging window.

The robot will move the TCP along the world coordinate axes (see Figure 20).

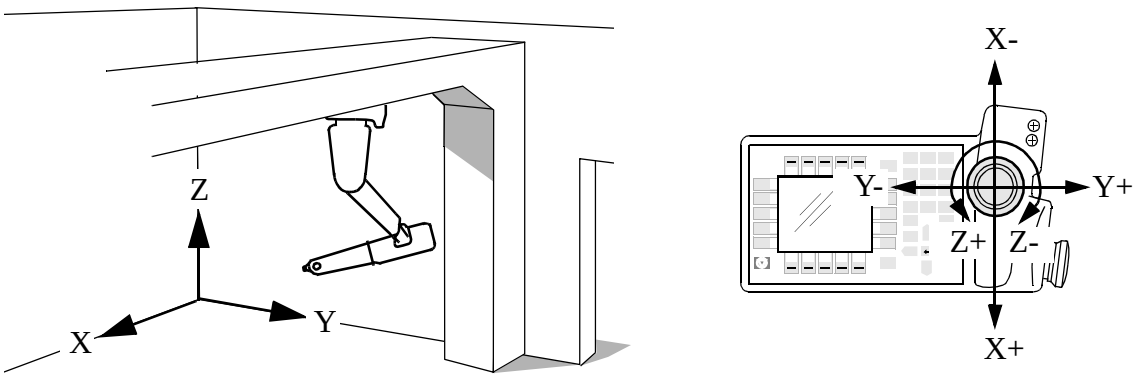


Figure 20 TCP movement is independent of the robot mounting.

2.7 Using a stationary tool

If a stationary TCP is active, the work object will move in accordance with the chosen coordinate system.

2.8 Jogging the robot axis-by-axis

• Choose axis-by-axis movement by setting the motion keys (see Figure 21).

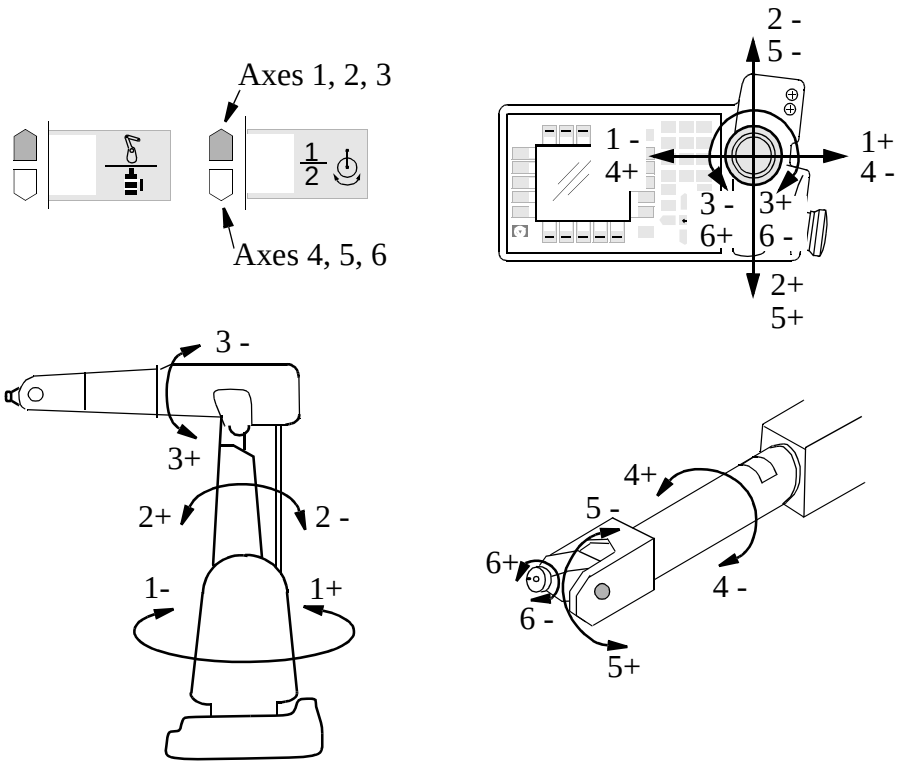


Figure 21 Specify the axes you want to move by setting the keys as above.

Only the axis affected by the joystick deflection moves, which means that the TCP does not move linearly.

2.9 Incremental movement

Incremental movement is used to adjust the position of the robot exactly. This means that each time the joystick is moved, the robot moves one step (increment). If the joystick is deflected for one or more seconds, a sequence of steps, at a rate of 10 steps per second, will be generated as long as the joystick is deflected.

• Select the field ***Incremental*** (see Figure 22).

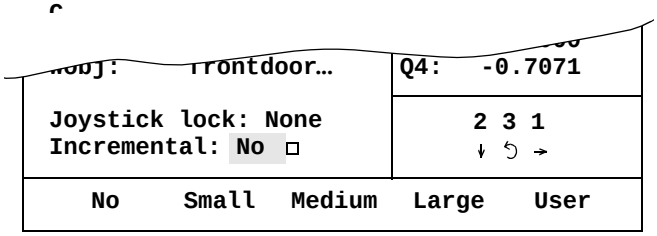


Figure 22 Specify the incremental step size in the field Incremental.

• Specify the size of the steps using the function keys.

- ***No:*** Normal (continuous) movement
- ***Small:*** Approx. 0.05 mm or 0.005 degrees per joystick deflection
- ***Medium:*** Approx. 1 mm or 0.02 degrees per joystick deflection
- ***Large:*** Approx. 5 mm or 0.2 degrees per joystick deflection
- ***User:*** User defined increments

You can also use the key --- on the teach pendant to turn incremental movement on and off.

If you want to specify the sizes of the user defined increments:

• Choose **Special: Increments**

A dialog box appears displaying the values of the increments for the different motion types (see Figure 23).

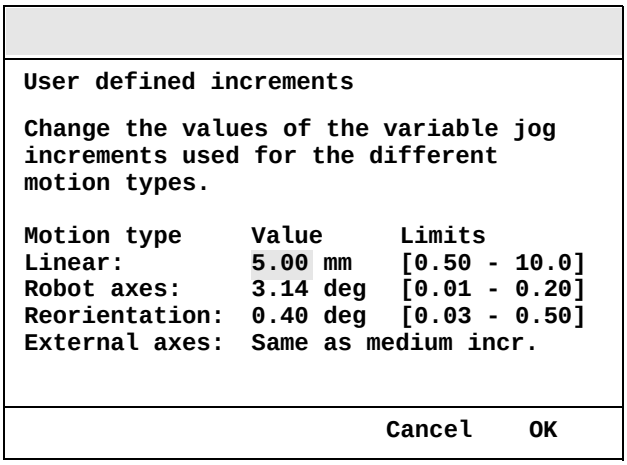

User defined increments
Change the values of the variable jog increments used for the different motion types.
Motion type Value Limits
Linear: 5.00 mm [0.50 - 10.0]
Robot axes: 3.14 deg [0.01 - 0.20]
Reorientation: 0.40 deg [0.03 - 0.50]
External axes: Same as medium incr.
Cancel
OK

Figure 23 The dialog box for specifying the user defined increments.

• Change the applicable value(s) and press ***OK*** to confirm.

2.10 Jogging an unsynchronised axis

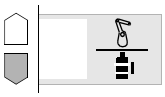
If the robot or an external unit is unsynchronised, it can only move using one motor at a time.

The working range is not checked, which means that the robot can be moved until it is stopped mechanically.

3 Jogging External Axes

3.1 Choosing external units

If you wish to use more than one external unit, those units must be chosen from the Jogging window.

• Set the motion key  to choose external units.

• Select the field ***Unit*** (see Figure 24).

• Using the function keys, choose a unit.

If you are using more than 5 external units and you cannot find the one you want in the function key dialog, press Enter ↵ and select the desired unit from the new dialog.

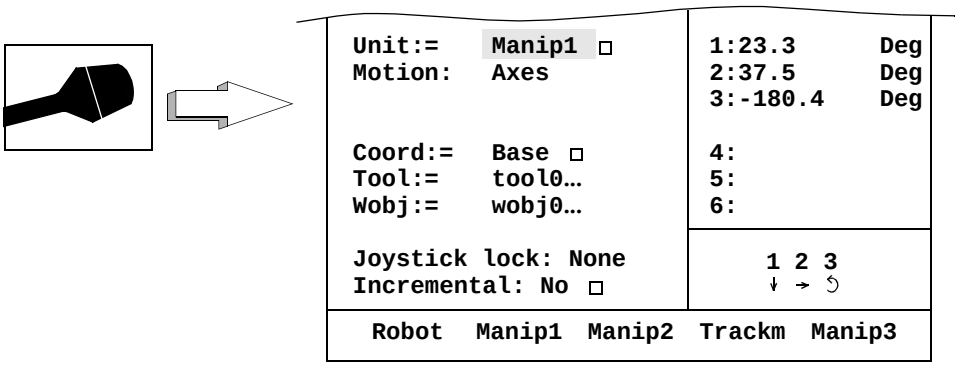


*Figure 24 Specify the unit to be jogged in the **Unit** field.*

From this stage onwards, the key can be used to toggle between the external unit that was last chosen and the robot.

3.2 Jogging external units axis-by-axis

• Choose the desired axis group using the motion keys (see Figure 25). If more than one external unit is used, see *3.1 Choosing external units*.

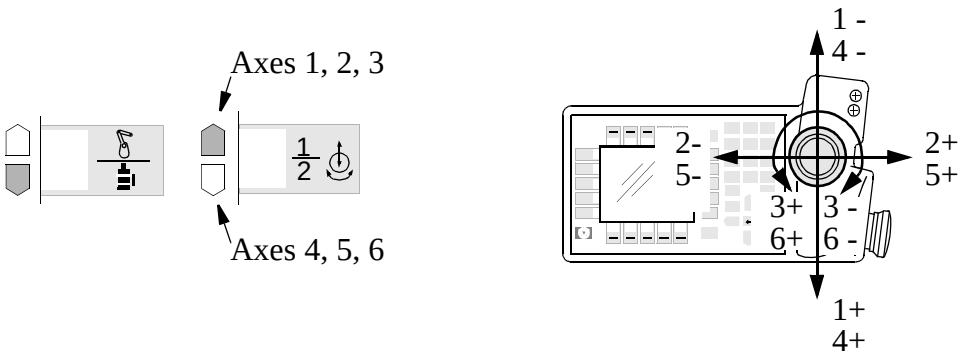


Figure 25 Specify the external axes you want to move by setting the keys as above.

3.3 Jogging external units coordinated

If an axis is coordinated with the robot (defined by the chosen work object), the robot also moves when it is jogged. The TCP, however, will not move in relation to the work object.

If you want to jog the unit uncoordinated, choose a work object which is not connected to a coordinated unit, e.g. *wobj0*, in the field ***Wobj***.

Jogging

CONTENTS

Page

1 General 3
1.1 The Inputs/Outputs window 3
1.2 Choosing an I/O list 4
1.3 Defining the *Most Common* I/O list 4
2 Changing Signal Values 6
2.1 Changing the value of a digital output 6
2.2 Changing the value of an analog output signal or a group of output signals 6
3 Displaying Information 7
3.1 To display information on a given signal 7
3.2 To display a chart of all digital signals of a unit 7
3.3 To print an I/O list 8

Inputs and Outputs

1 General

The robot can be equipped with both digital and analog signals. The signals are named and configured in the system parameters. They can also be assigned various system actions, e.g. program start.

In addition to this the robot can communicate with printers and computers via serial channels and Ethernet.

1.1 The Inputs/Outputs window

• Press the Inputs/Outputs key to open the window.

The window displays a list of appropriate signals or units. It also provides information on the values of the signals. See the example in Figure 1.

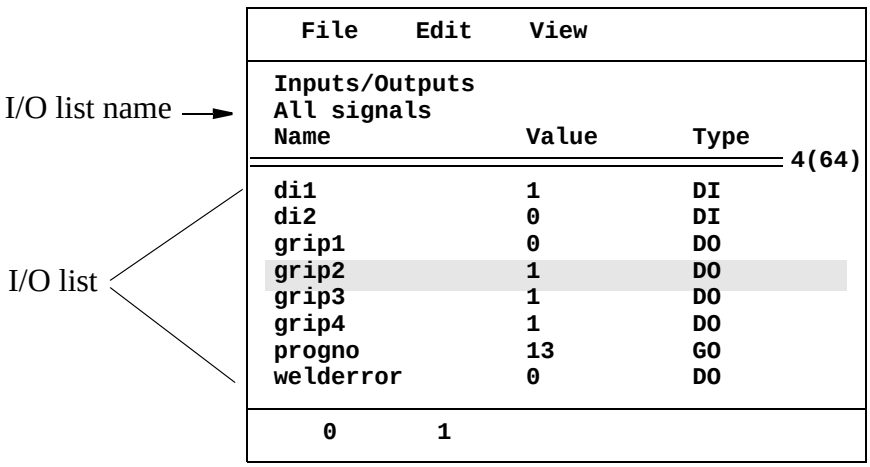


Figure 1 The Inputs/Outputs window displays a list of selected signals or I/O units.

The information displayed in the window is automatically updated every other second.

1.2 Choosing an I/O list

• You can decide which signals you want to look at by choosing any of the lists from the **View** menu:

List name	Information in the list
Most Common	The value of the most important (most used) signals. This list can be customised to suit any robot installation.
All signals	The value of all signals.
Digital In	The value of all digital input signals.
Digital Out	The value of all digital output signals.
Analog	The value of all analog input and output signals.
Groups	The value of all groups of digital signals.
Safety	The value of all safety signals.
Units	The type and address of all I/O units.
I/O Unit: *name*	The value and position of all signals of a unit. To look at this list: • Choose **View: Units**. • Select the desired unit and press Enter ↵.
Group: *name*	The value and position of all signals in a signal group. To look at this list: • Choose **View: Groups**. • Select the desired unit and press Enter ↵

1.3 Defining the *Most Common* I/O list

You can obtain an easy-to-access list of your most frequently-used signals by specifying the contents of the *Most Common* list.

• Choose **File: Preferences**.

All signals will be displayed. Those included in the *Most Common* list will be marked with an *x* to the left of their names (see Figure 2).

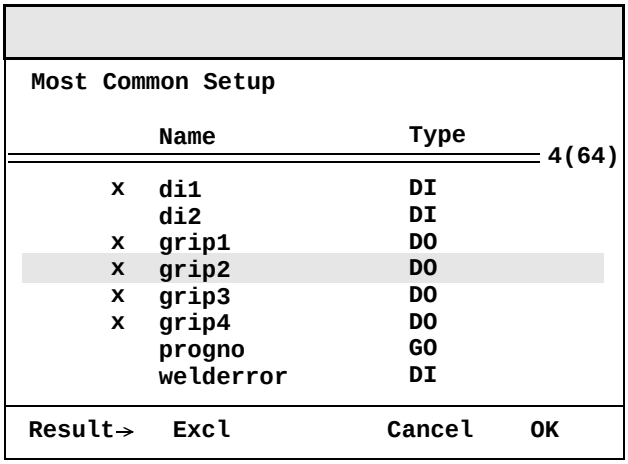


Figure 2 You specify the signals to be included in the list in the Most Common Setup dialog box.

- To add a signal, select an appropriate signal and press ***Incl.***
  This signal will then be marked with an *x* to the left of its name.
- To remove a signal, select an appropriate signal and press ***Excl.***
  This signal will remain in the window, but the *x* to the left of the signal name will disappear.
- Press ***Result***.

The signals included in the *Most Common* list will then be displayed (see Figure 3).

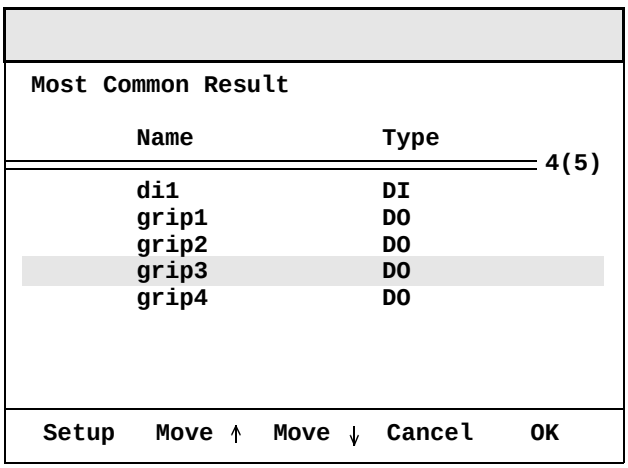


Figure 3 The order of the signals in the list can be specified in the Most Common Result dialog box.

- You can change the order of the signals using the ***Move***-keys. The selected signal moves one step at a time.
- Define the signal and press ***OK***; if you want to return to the *Most Common Setup* dialog box press ***Setup*** first.

2 Changing Signal Values

Robot equipment may be affected (e.g. start to move or fall off) if you change the value of a signal.
Before you do so, make sure that no-one is in the safeguarded space around the robot. Incorrect operation can injure someone, or damage the robot or other equipment.

2.1 Changing the value of a digital output

- Select the digital output.
- Choose the desired value using the function keys (see Figure 4).

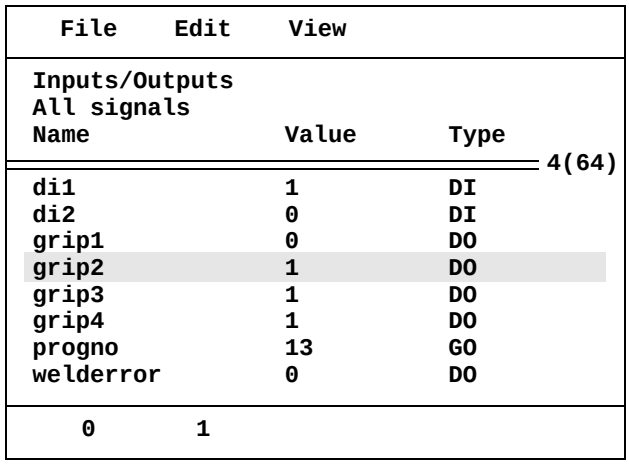


Figure 4 You can change the value of a digital output directly using the function keys.

2.2 Changing the value of an analog output signal or a group of output signals

- Select the signal and press ***Change*** (see Figure 5).

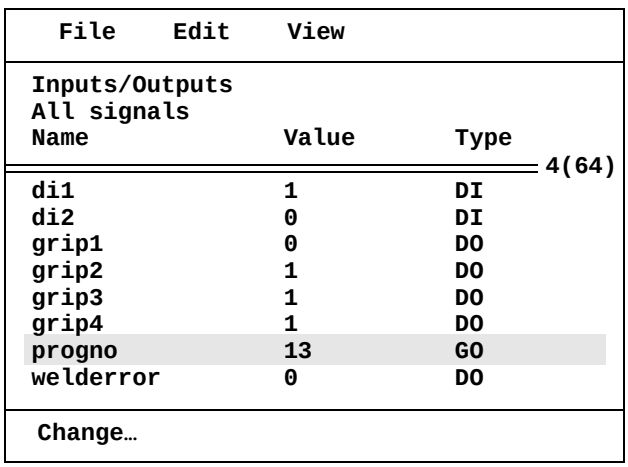


Figure 5 You can change a group of outputs or an analog output signal by choosing Change and entering a value using the numeric keyboard.

A dialog box will appear, which you can use to enter an arbitrary value.

• Specify the desired value using the numeric keyboard and press ***OK***.

You can also change the value of a group of output signals, signal by signal, by pressing Enter ↵ and changing the signals one by one.

3 Displaying Information

3.1 To display information on a given signal

• Select the desired signal and press Enter ↵.

The following information will be displayed:

- the signal name
- the signal type
- the value of the signal
- the physical connection
- cross-connections (if any)
- etc.

• Press ***OK*** when ready.

3.2 To display a chart of all digital signals of a unit

• Call up the unit list by choosing **View: Units**.

• Select the desired unit and press the ***State*** function key.

The values of all digital signals related to the selected I/O unit will appear on the display (see Figure 6). The values of the signals are indicated by 1 or 0, where, for example, 1 is equivalent to +24 V and 0 is equivalent to 0 V.
An “x” means that the signal is not mapped (cannot be used in a program).
A “?” means that the signal value cannot be read.

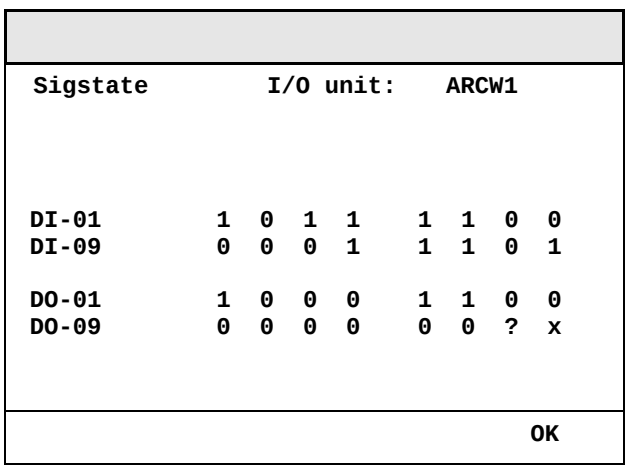


Figure 6 The value of all the digital values of an I/O unit are displayed together on a signal chart.

• Leave the signal chart by pressing ***OK***.

3.3 To print an I/O list

- Select the desired I/O list from the **View** menu.
- Choose **File: Print**.

A dialog box will be displayed (see Figure 7).

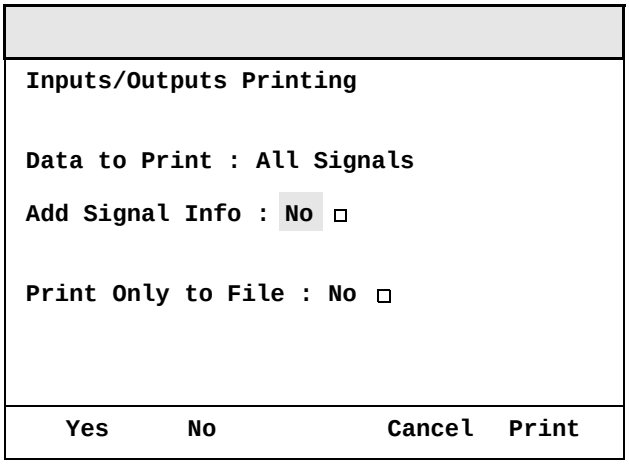


Figure 7 You can specify the extent of information and the destination.

- In the field, ***Add Signal Info***, specify how much you want to print:
  - Press ***No*** to print the list.
  - Press ***Yes*** to print other information about the signals, such as their configuration.
- Select the destination in the field, ***Print Only to File***:
  - Press ***No*** to output to the printer connected to the robot.
  - Press ***Yes*** to save the list in a file. An additional line with the filename will be displayed. If you want to change the filename, select it and press Enter ↵.
- Start the print-out by pressing ***Print***.
- Press ***OK*** to confirm.

CONTENTS

Page

1 Creating a New Program 5
1.1 What is a program? 5
1.2 The Program window 6
1.3 Creating a new program 6
1.4 Loading an existing program 7
2 Defining Tools and Work Object 7
3 Creating New Routines 8
3.1 What is a routine? 8
3.2 The *Program Routines* window 9
3.3 Creating a new routine 10
3.4 Duplicating a routine 11
4 Creating new instructions 12
4.1 Choosing a routine 12
4.2 The *Program Instr* window 12
4.3 What is an instruction? 13
4.4 Getting more information about an instruction 14
5 Programming 14
5.1 Choosing from the instruction pick list 15
5.2 Adding an instruction 16
5.3 Expressions 18
5.4 Moving and copying instructions 21
6 Running Programs 21
6.1 Program execution 21
6.2 The *Program Test* window 22
6.3 Choosing the speed correction 22
6.4 Choosing the execution mode 23
6.5 Starting program execution 24
6.6 Stopping program execution 25
6.7 Where will the program start? 25
6.8 Simulating wait conditions 27
7 Saving and Printing Programs 28
7.1 Saving the program on diskette or some other type of mass memory 28
7.2 Printing a program from the robot 29
7.3 Printing a program using a PC 29
8 Changing the Program 29
8.1 Selecting an instruction or an argument 30

8.2 Modifying the position in a positioning instruction 31
8.3 Tuning position during program execution 31
8.4 Changing an argument 33
8.5 Adding optional arguments 34
8.6 Changing the structure of an IF, FOR or TEST instruction 35
8.7 Changing the name or declaration of a routine 35
8.8 Deleting an instruction or an argument 36
8.9 Deleting a routine 36
8.10 Undo latest action 36
9 Special Editing Functions 37
9.1 Search & replace 37
9.2 Mirroring 39
10 Creating Data 45
10.1 What is data? 45
10.2 The *Program Data* window (used to manage data) 45
10.3 Creating new data 47
10.4 Creating new array data 48
10.5 Duplicating data 50
10.6 Storing position data using the robot 50
10.7 Routine data 50
11 Changing Data 50
11.1 Viewing and possibly changing the current value 50
11.2 Changing data names or declarations 51
11.3 Deleting data 52
12 Error Handling 52
13 Using Modules 54
13.1 What is a module? 54
13.2 Choosing modules 55
13.3 Creating a new module 56
13.4 Changing the name or declaration of a module 56
13.5 Reading a program module from diskette or some other type of mass memory. 57
13.6 Deleting program modules from the program 57
13.7 Listing all routines in all modules 57
13.8 Duplicating a routine from one module to another 58
13.9 Listing all data in the current module 58
13.10 Duplicating data from one module to another 58

13.11 Saving modules on diskette or some other type of mass memory 58
13.12 Calling up the complete module list 59
14 Preferences 60
14.1 Defining the *Most Common* instruction pick list 60
14.2 Default data Global/Local 61
14.3 Defining programming rule for robot positions 62

Programming and Testing

1 Creating a New Program

1.1 What is a program?

A program consists of instructions and data, programmed in the RAPID programming language, which control the robot and peripheral equipment in a specified way.

The program is usually made up of three different parts:

- a main routine
- several subroutines
- program data.

In addition to this, the program memory contains system modules (see Figure 1).

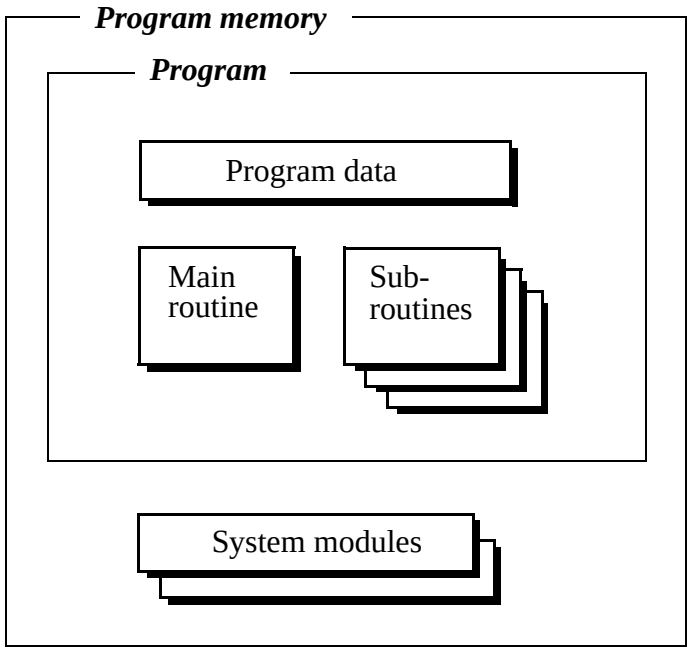


Figure 1 The program instructions control the robot and robot peripherals.

The main routine is the routine from which program execution starts.

Subroutines are used to divide the program up into smaller parts in order to obtain a modular program that is easy to read. They are called from the main routine or from some other routine. When a routine has been fully executed, program execution resumes at the next instruction in the calling routine.

Data is used to define positions, numeric values (registers, counters) and coordinate systems, etc. Data can be changed manually, but it can also be changed by the program; for example, to redefine a position, or to update a counter.

An *instruction* defines a specific action that is to take place when the instruction is executed; for instance, moving the robot, setting an output, changing data or jumping within the program. During program execution, the instructions are executed one at a time, in the order in which they were programmed.

System modules are programs that are always present in the memory. Routines and data related to the installation rather than the program, such as tools and service routines, are stored in system modules.

1.2 The Program window

All program and testing is performed using the Program window.

• Press the Program key to open the window.

The Program window is actually made up of a number of different windows. These can be chosen from the **View** menu.

Window title	Used to:
Program Instr	Program and change program instructions
Program Routines	Choose or create new routines
Program Data	Create or change data
Program Data Types	Choose data of a specific type
Program Test	Test programs
Program Modules	Choose or create new modules

1.3 Creating a new program

• Choose **File: New**.

If the robot is already loaded with a program which has not been saved, a dialog box appears and you will be asked whether you want to save it or not. Then choose **File: New** again.

• Specify the new name of the program in the dialog box that appears.
(See Chapter 5, Starting up - *Entering text using the teach pendant*, in this manual for how to handle the text editor.)

• Choose ***OK*** to confirm.

A program with only one empty main routine is created.

1.4 Loading an existing program

• Choose **File: Open**.

A dialog box appears, displaying all programs in the current directory (see Figure 2).

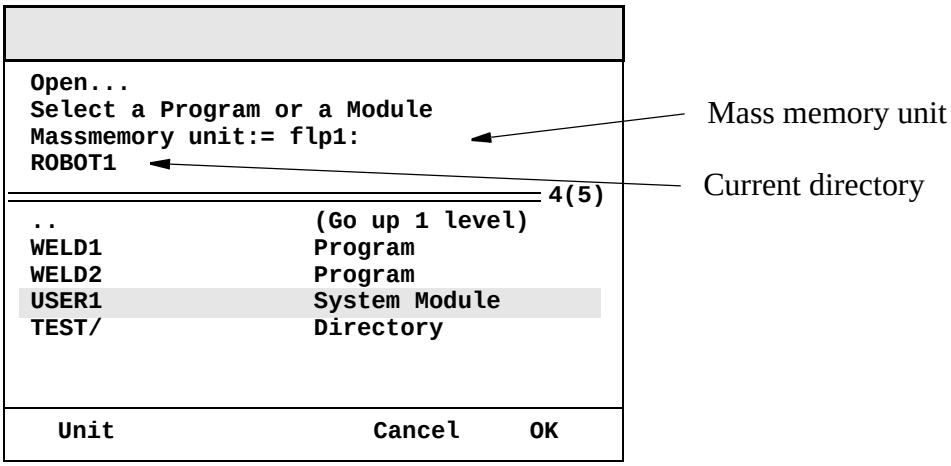


Figure 2 The dialog box used to read programs.

• If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

• Select the desired program. Move up or down in the directory by using either '. .' (up), or the desired directory (down) and press Enter ↵.

• Choose ***OK*** to confirm.

When a program is already loaded into the system, but has not been saved, and you wish to open another program, a dialog box appears and you will be asked whether you want to save the old program or not.

Tip If there is an error in the program, this error will be displayed if you choose **File: Check Program**.

Note When a program is loaded into the robot, it requires about three times as much memory compared with the size of the file on diskette.

2 Defining Tools and Work Object

Before starting any programming work, it is essential that you define the tools, work objects, and other coordinate systems that you intend to use. The more accurately you do this, the better the results you will obtain.

See Chapter 10, Calibration.

3 Creating New Routines

3.1 What is a routine?

Before you start to program, you should think out the structure of your program:

- The program should be divided into several subroutines to get a more readable program.
- Instruction sequences that recur frequently in the program, such as gripper handling, form their own routines.

Figure 3 illustrates an example of a simple program; the robot takes parts to and from a machine. Figure 4 illustrates the structure of this program.

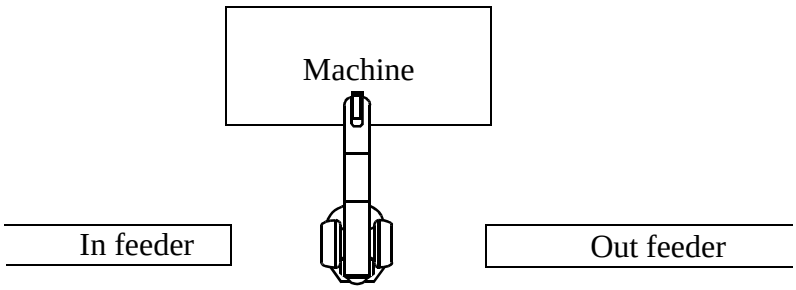


Figure 3 The robot gives a part to a machine which then processes it.

First, the robot fetches a part from the In feeder and places it in the machine where the part is processed. Then, when this has been done, the robot takes the part and places it on the Out feeder.

The main routine is built up of a number of routine calls which reflect the robot work cycle (see Figure 4).

As the gripper grips and releases parts several times during the program run, it is best to set up separate routines for this, which can be called from different places in the program.

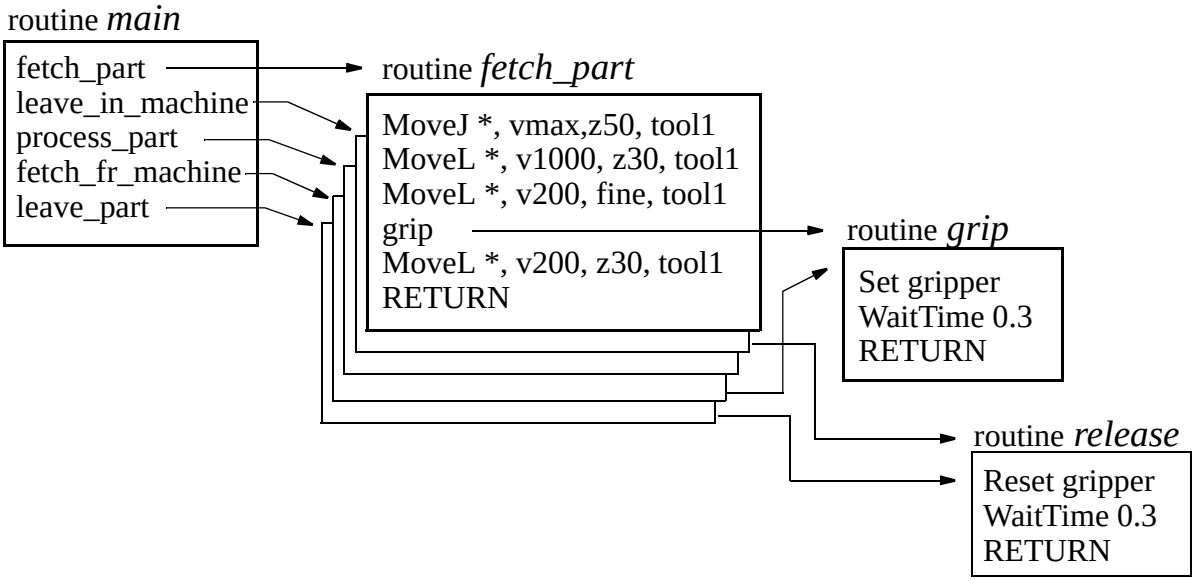


Figure 4 For more information about this example, see Chapter 17, Program Examples.

There are three types of routines: procedures, functions and trap routines.

A *procedure* could be described as a number of instructions that perform a specific task, such as welding a part or changing a tool.

A *function* returns a value and, for example, is used to displace a position or read an input.

A *trap routine* is used to deal with interrupts.

A routine comprises four parts: declarations, data, instructions and an error handler (see Figure 5).

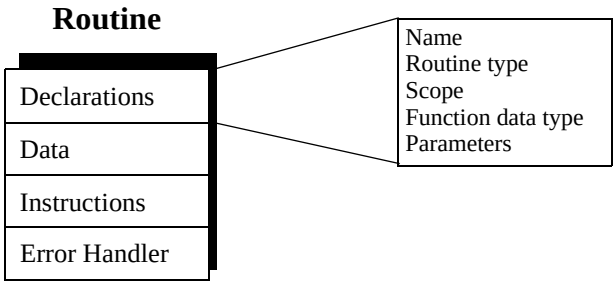


Figure 5 A routine comprises declarations, routine data, instructions and an error handler.

The *declaration* specifies routine parameters, among other things. These are used to make the routine more generally applicable. A routine that, for example, moves the robot a given distance in the direction of the tool can have that distance as a parameter. This routine can then be called using different distances and thus can be used to move the robot different distances.

The *error handler* takes care of automatic error handling (see *Error Handling* on page 52).

3.2 The *Program Routines* window

• Choose **View: Routines** to open the window.

The window displays routines and, if there is a function present, also the type of data returned for that function (see Figure 6).

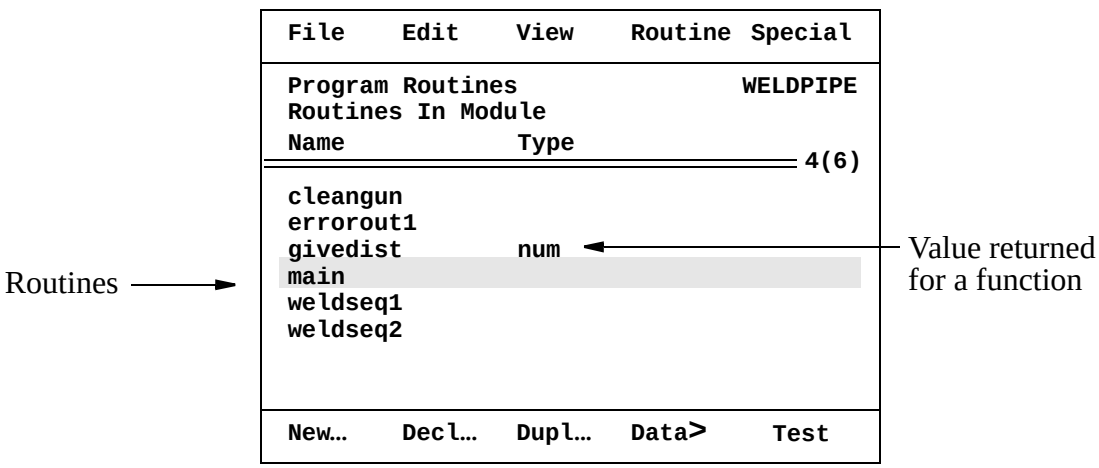


Figure 6 The Program Routines window displays all routines in the program.

3.3 Creating a new routine

• Open the *Program Routines* window by choosing **View: Routines**.

• Press the function key ***New.***

A dialog box appears, displaying the name of the routine (see Figure 7). The name is set to *routineN*, where *N* is a number incremented each time a routine is created.

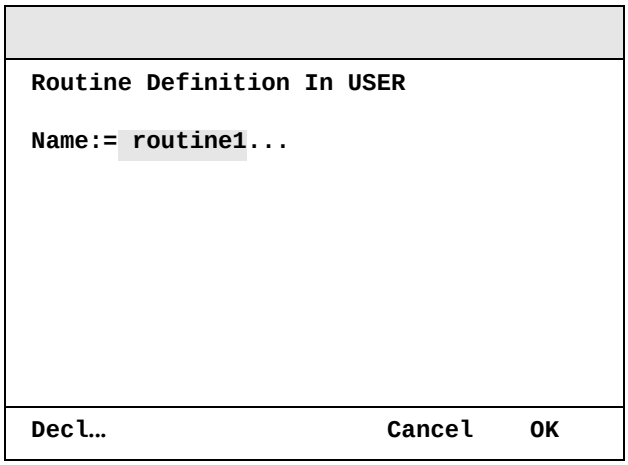


Figure 7 A new routine is created.

• Change the name by pressing Enter ↵ and specify a new name.

If you want a normal subroutine (procedure), without parameters, you should finish here by pressing ***OK***. In other cases, the characteristics of the routine must be defined.

• Press the function key ***Decl***.

• Change the characteristics of the routine by selecting the appropriate field, then:

- Press Enter ↵ and specify the desired alternative in the dialog box that appears on the display (fields marked with ...).
- Choose an alternative using the function keys (fields marked with ▫).

Field	Description
Name	The name of the routine (a maximum of 16 characters)
Type	Specifies whether the routine is to be a procedure (***Proc***), a function (***Func***) or a trap routine (***Trap***)
In Module	The module in which the new routine will be used
Data type	The return value for the data type (only specified for functions)

If the routine is not to include any parameters, you can terminate the definition by pressing ***OK.*** In other cases, the parameters must also be defined.

• Select the parameter list by pressing the List key.

• Add a parameter by pressing the ***New*** function key.

New parameters are placed after the selected parameter in the list. You can, however, move the parameter using ***Move*** ↑ (up one step) och ***Move*** ↓ (down); see Figure 8.

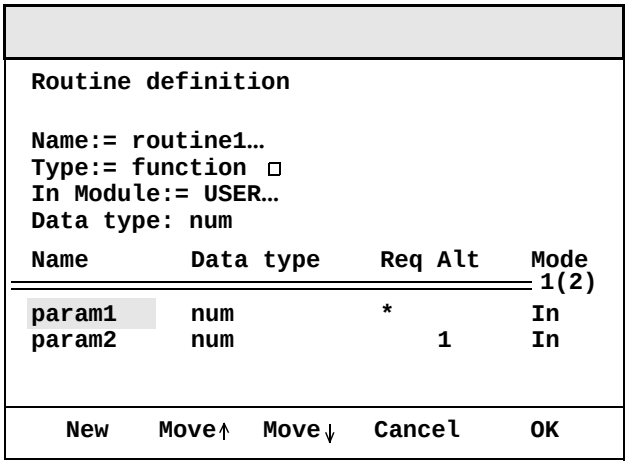


Figure 8 The dialog box used to define parameters.

- Change the name and characteristics of the parameter by selecting the appropriate field, then:
  - Press Enter ↵ and specify the desired alternative in the dialog box that appears.
  - Choose an alternative using the function keys.

Field	Description
Name	The name of the parameter (max. 16 characters).
Data type	The data type of the parameter.
Required	Specifies whether the parameter is compulsory (***Yes***) or can be omitted (***No***) at a call – marked with * in the list.
Alt	Non-compulsory parameters can be mutually exclusive, i.e. they cannot be used simultaneously in the instruction. To input the first of these parameters, press the function key ***First*** and to input the last one, press ***Tail***.
Mode	Specifies whether the parameter can only be read (***IN***) or whether it can be read and changed in the routine (***INOUT***).

- Add any additional parameters (maximum 16 parameters). To remove a parameter, select it and then press Delete ⌫.
- Choose ***OK*** to confirm.

Tip It is sometimes easier to create a new routine by duplicating and changing an existing one.

3.4 Duplicating a routine

- Choose **View: Routines**.
- Select the routine to be duplicated.

• Press the function key ***Dupl***.

• Specify the new name of the routine in the dialog box that appears.

• Choose ***OK*** to confirm the duplication.

This creates a new routine that contains the same data and instructions as the original routine.

4 Creating new instructions

4.1 Choosing a routine

• Choose **View: Routines**.

• Select the routine to be programmed and press Enter ↵.

To call up the main routine

• Choose **View: Main Routine**.

To call up a routine that can be selected from the list of instructions

• Select the routine that you want to look at.

• Choose **View: Selected Routine**.

4.2 The *Program Instr* window

• Choose **View: Instr** to open the window.

If you are in the *Program Test* or *Program Data* window, you can press the function key ***Instr*** instead.

The instructions for the current routine are displayed in the window (see Figure 9).

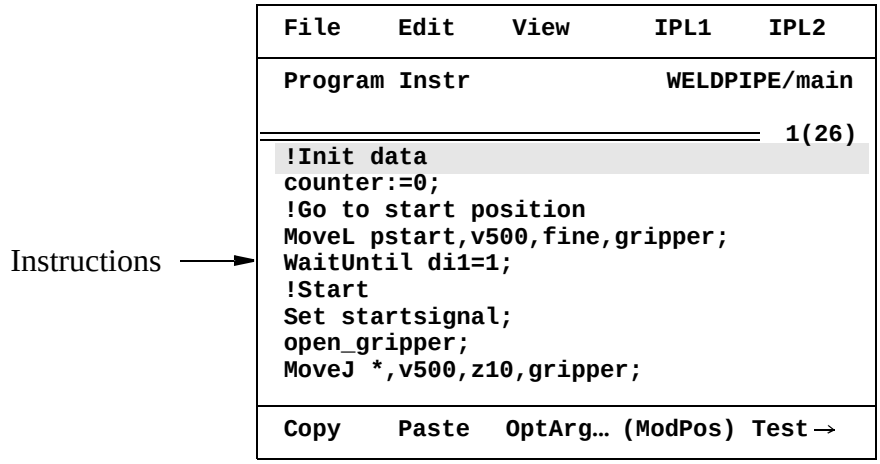


Figure 9 The Program Instr window is used for programming.

An instruction that does not fit into one line is only displayed in part. Arguments that lie outside the visible area are moved successively inwards from the right when the various arguments are selected.

4.3 What is an instruction?

An instruction defines a specific task that is to be carried out when the instruction is executed, for example:

- Moving the robot
- Setting an output
- Changing data
- Jumping within the program.

Instructions comprise an instruction name and a number of arguments. The name specifies the principal task of the instruction and the arguments specify the characteristics.

An argument may be either compulsory (required) or optional. Optional arguments may be omitted, and are specified by the name of the argument and its value, if it has one. For example:

Instruction	Meaning
MoveL p1,v500,fine,tool1	Moves the TCP linearly to the position *p1*. The arguments, *v500*, *fine* and *tool1*, specify the current speed, position accuracy and tool.
SetDO do2,1	Sets the output *do2* to *1*.
SetDO \SDelay:=0.5,do2,1	Sets the output *do2* to *1* with a delay of *0.5* seconds. *\SDelay* is an optional argument, *do2* and *1* are compulsory.

An argument that does not have a specified value is marked with <...>.
Programs that contain such instructions (i.e. incomplete instructions) can be executed, but program execution stops when that type of instruction occurs.

Arguments can be specified as:

- numeric values, e.g. *1*
- string values, e.g. *“Waiting for machine”*
- data, e.g. *reg2*
- function calls, e.g. *Abs(reg2)*
- expressions, e.g. *reg2 + reg3 / 5*.

4.4 Getting more information about an instruction

• Select the desired instruction and press Enter ↵.

The dialog box shows the names of the arguments (see Figure 10).

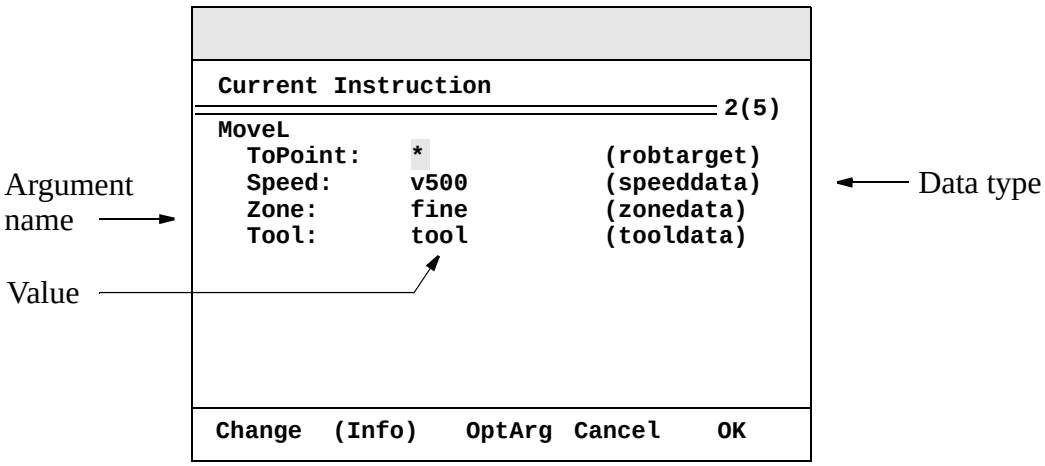


Figure 10 The name, value and data type of each argument is displayed.

• If you wish to change an argument, choose ***Change*** or press Enter ↵. See *Changing an argument* on page 33 for more information.

• If you wish to add or remove an optional argument, choose ***OptArg***. See *Adding optional arguments* on page 34 for more information.

• Choose ***OK*** to exit the dialog.

5 Programming

In this chapter you will find descriptions for general handling of the various instructions in a program - moving, copying or adding. For details about programming the most common instructions please see the next chapter in this manual, *9 The programming language RAPID*.

For other instructions see the RAPID Reference Manual.

5.1 Choosing from the instruction pick list

You can choose instructions by selecting an appropriate instruction in an instruction pick list (IPL). Although most of these pick lists are fixed, some can be user-defined. This means that you can place the instructions you use most in the same pick list (see *Defining the Most Common instruction pick list* on page 60).

The following pick lists are available:

From the IPL1 menu

Name	Contains
Common	Some of the most-commonly used instructions
Prog. Flow	Instructions that control the program flow
Various	E.g.':=' and wait
Motion Settings	Instructions that affect movements
Motion & Process	Motion instructions
I/O	I/O instructions
Communicate	Communication instructions
Interrupts	Instructions that handle interrupts
Error Recovery	Instructions that handle errors
System & Time	Date and time instructions
Mathematics	Arithmetic instructions

From the IPL2 menu

Most Common 1	User-defined
Most Common 2	User-defined
Most Common 3	User-defined
Motion Set Adv.	Advanced Motion setting instructions
Motion Adv.	Advanced Motion instructions
Ext. Computer	Communication Ware instructions
Service	Service instructions

- Call up one of the instruction pick lists in the **IPL1** or **IPL2** menu.
- Call up the instruction pick list that was used most recently by pressing **Edit: Show IPL**. If the pick list contains more than 9 instructions you can scroll up/down in the list using 9 on the numeric keyboard.
- Change to the previous or next pick list by selecting the pick list ( ) and pressing PreviousPage or NextPage . You can also choose 0 to go directly to the next page.
- Remove the instruction pick list by choosing **Edit: Hide IPL**.

5.2 Adding an instruction

If a new instruction is added, it is placed after the instruction that is selected.

If the selected instruction is first in a routine, or in a compound instruction (IF, FOR, WHILE or TEST), you can choose whether you want the new instruction to be placed before or after the instruction (by means of a question). However, if there is only one instruction in the routine, or in the compound instruction, new instructions will always be added after the selected one.

- Select the place where you want the new instruction to be added.
- Call up one of the instruction pick lists by choosing the appropriate pick list from the **IPL1** or **IPL2** menu. If you want to call up the instruction pick list that was used most recently, choose **Edit: Show IPL**.

The pick list will be displayed on the right hand side of the window (see Figure 11).

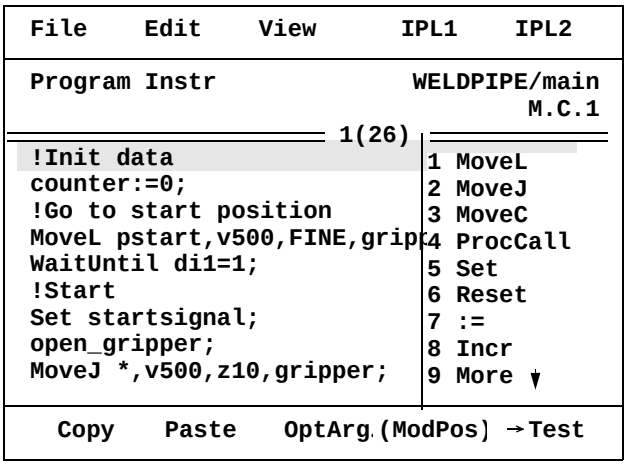


Figure 11 The instructions are chosen from an instruction pick list.

- Choose the desired instruction using one of the following alternatives:
  - Using the numeric keyboard, press the number displayed in front of the appropriate instruction in the pick list.
  - Select the pick list by pressing the List key. Then, select the desired instruction and press Enter .
  - Use 0 on the numeric keyboard to scroll down to the lower part of the pick list or to the next pick list.

If the instruction has no arguments, or if these are automatically set, the instruction is ready for use right away.

If the instruction has arguments that cannot be automatically set, a dialog box will appear in which you can specify the value of the instruction arguments. The argument is marked with a ”?” in front of it (see Figure 12).

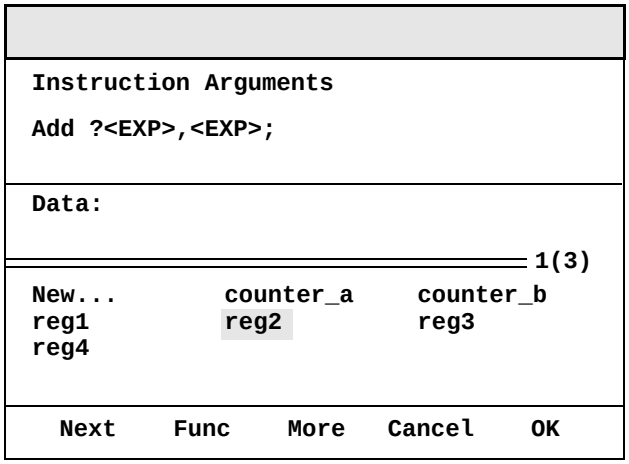


Figure 12 The dialog box used to define arguments. In this example, the instruction Add is programmed.

The argument can now be defined in four different ways:

- by entering a numeric value directly using the numeric keyboard
- by choosing data in the lower part of the dialog box

  New, the first alternative in the list, is used when you want to create new data and refer to it. If you choose ***New***, you define new data as described in *Creating Data* on page 45.

- by choosing a function; press the function key ***Func*** and select the desired alternative from the list

  A new dialog box that can be used to program arguments appears, like the one in Figure 12. Specify the function argument in the same way as you specified the instruction argument. Use the function key ***Skip*** to delete optional arguments that are not to be included.

- by entering an expression by pressing ***More***.

  For more information, see *Programming an expression* on page 19.

• Choose ***Next*** to change the next argument.

• Choose ***OK*** to confirm.

Optional arguments that are not included at the start can be inserted, see *Adding optional arguments* on page 34.

The structure of an IF, FOR or TEST instruction can be changed, see *Changing the structure of an IF, FOR or TEST instruction* on page 35.

5.3 Expressions

What is an expression?

An expression is used as an argument of an instruction and can have an arbitrary number of components.

There are three different types of expressions:

- logical expressions;
  these have the value true/false and are used together with tests, e.g.
  IF reg1=5 AND reg2 >10 .....
  IF di1 = 1 .....
- arithmetic expressions;
  these have a numeric value and are used together with calculations, e.g.
  reg1 = reg2 + 3 * reg5
  reg1 = reg2 + 1
- strings, e.g.;
  TPWrite “Producing”

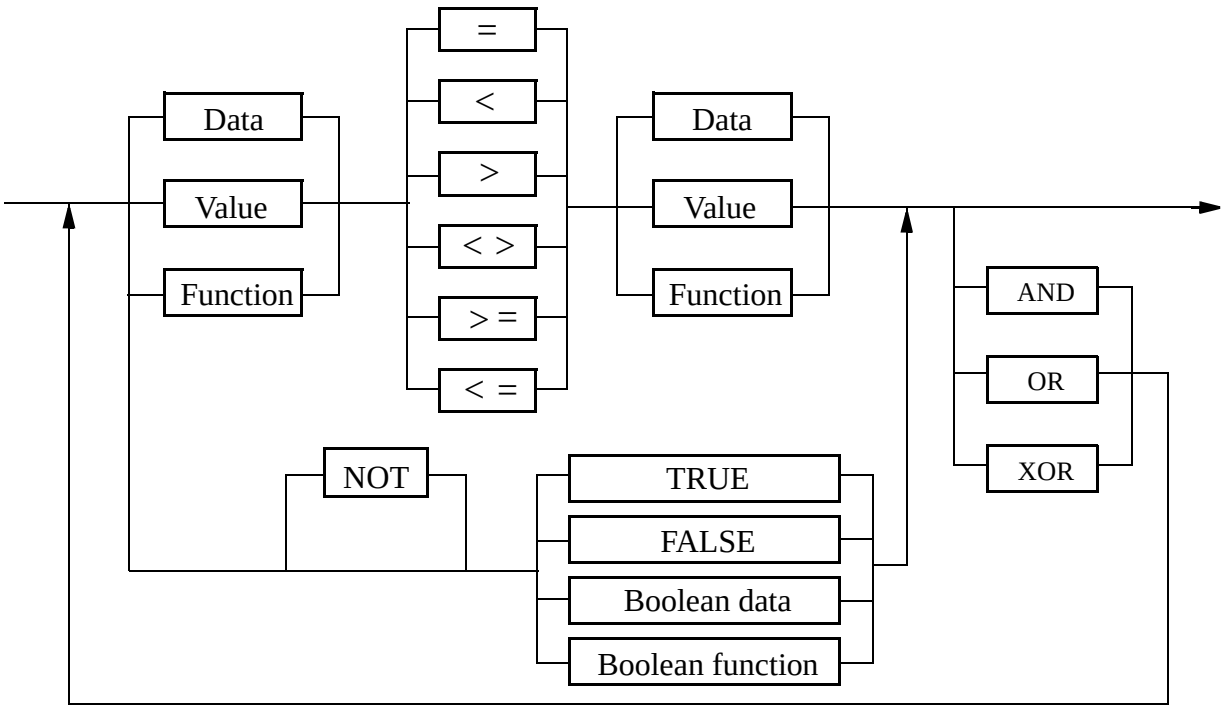


Figure 13 Logical expression.

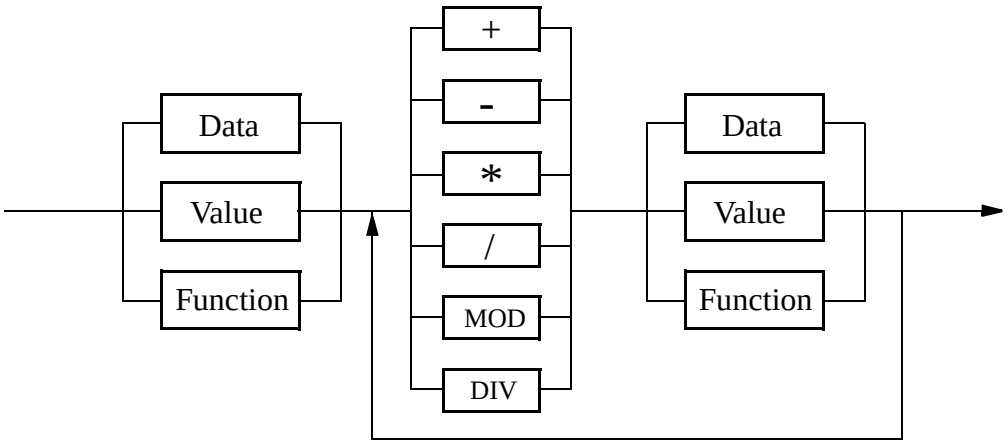


Figure 14 Arithmetic expression.

Programming an expression

Expressions are programmed by pressing the function key ***More*** in the instruction argument dialog box (see Figure 12).

Expressions can be entered or changed directly in the upper part of the dialog box (see Figure 15) by doing any of the following:

- move the cursor to the left or right using ArrowLeft ⟨ or ArrowRight ⟩ ;
- delete what is marked by the cursor by pressing Delete ⌫ ;
- add digits in front of the cursor using the numeric keyboard.

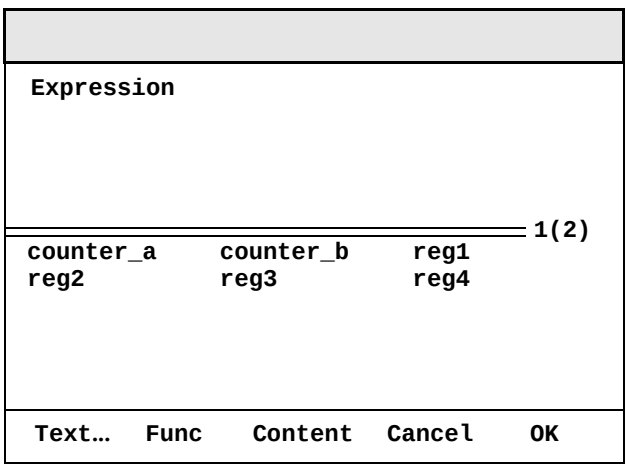


Figure 15 The Expression dialog box.

Data, functions and operators can be selected in the lower part of the dialog box. Press ☰, if needed, select the desired alternative and press Enter ↵ .

Enter text by pressing ***Text***. A dialog box appears in which, using the function keys and the numeric keyboard, you can enter text.

If the desired information is not in the lower part, press one of the function keys ***Data***, ***Func*** or ***Content*** first.

- ***Data*** gives a list of all user-defined data of the selected data type
- ***Func*** gives a list of all functions of the selected data type
- ***Content*** gives an intermediate dialog where data of a new data type can be chosen in the same way as the IF instruction, for example. You can also choose to view user-defined or system-defined data, or both. √ denotes the current choice (see Figure 16).

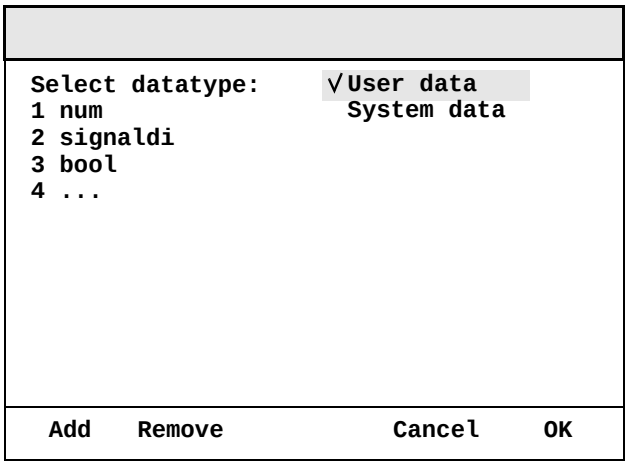


Figure 16 Dialog box for choosing data types.

Editing an expression

Move the cursor using the arrow keys. The content of the list will change so that it corresponds to that selected. The function key ***Content*** changes to ***Insert*** (see Figure 17).

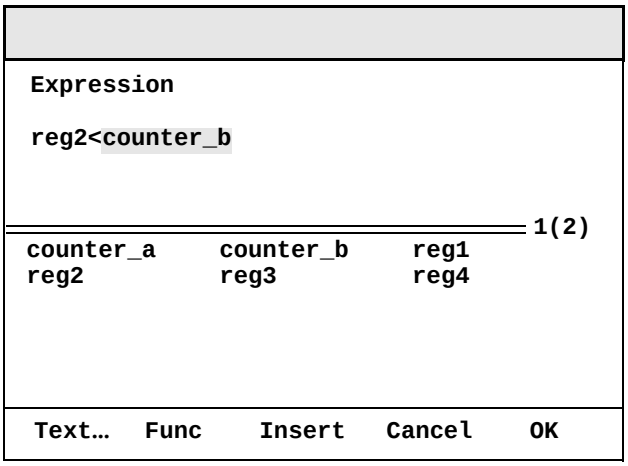


Figure 17 The dialog box for editing an expression.

Replace what has been selected by selecting the desired choice in the lower part of the box and pressing Enter ↵ .

You can make an addition to an expression by pressing the function key ***Insert***. An underscored “blank” _ will then be inserted before the cursor and the function key ***Insert*** will change to ***Content*** (see Figure 18).

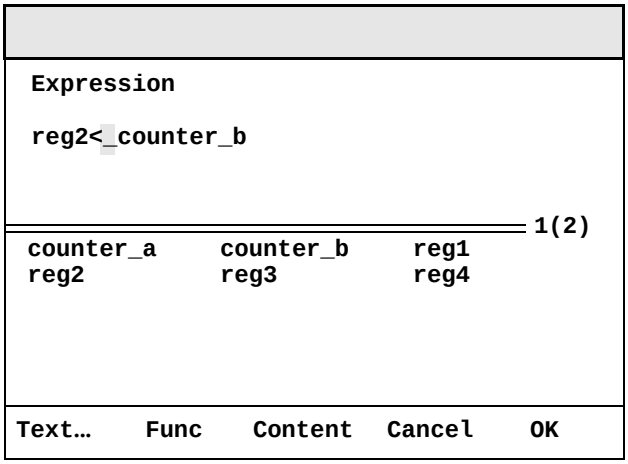


Figure 18 New data can be inserted into an expression.

5.4 Moving and copying instructions

- Select the instruction you wish to move or copy. To select several instructions, choose **Edit: Mark**.
- Choose **Edit: Cut** (move) or **Edit: Copy**.
- Indicate where you wish to add the new instructions.
- Choose **Edit: Paste**.

In the *Program Instr* window, copy and paste can also be selected using a function key.

6 Running Programs

6.1 Program execution

A program can be executed regardless of whether or not it is complete. Nonetheless, if program execution comes to an incomplete instruction, the program is stopped.

When the program is started, the robot checks that all references to data and routines are correct. If they are not, the fault is indicated and the program is not started. This check can also be performed by choosing **File: Check Program**. The first fault in the program is then indicated.

The program is usually started from the first instruction in the main routine, but can also be started from an arbitrary routine (procedure) with no parameters. A program that has been stopped is, unless otherwise specified, always started from the instruction last executed in the program.

6.2 The *Program Test* window

• Choose **View: Test**. When you are in the *Program Instr* or *Program Data* window, you can also press the function key ***Test***.

The section of the program that will be executed when you start the program is displayed in the window.

A program pointer keeps up with the program execution. This pointer is shown with » in the program list. Program execution normally continues from this point. However, if the cursor is moved to another instruction when the program is stopped, execution can be started from the position of the cursor (see Figure 19).

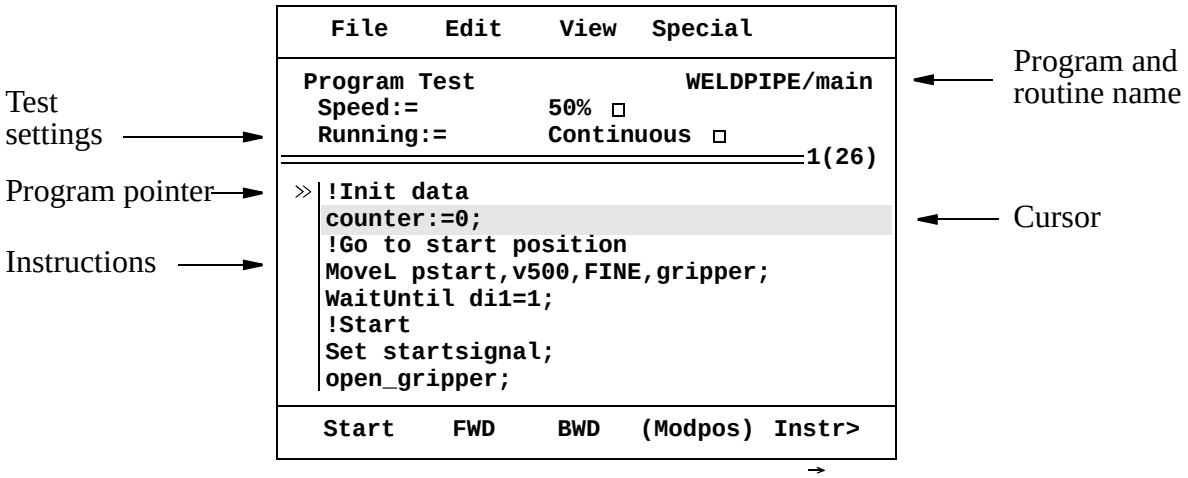


Figure 19 The Program Test window is used to execute a program.

If the robot is equipped with an arc-welding function, an extra field with the blocking status will be shown.

6.3 Choosing the speed correction

When the program is being tested for the first time, it is advisable to reduce the speed. A 50% speed correction means that the speed will be reduced to 50% of the programmed speed. On the other hand, when the robot is in manual mode with reduced speed, the speed is never more than 250 mm/s.

It is also possible to change the speed correction while the program is executing.

• Select the upper part of the window by pressing the List key (if it is not already selected).

• Select the field ***Speed*** (see Figure 20).

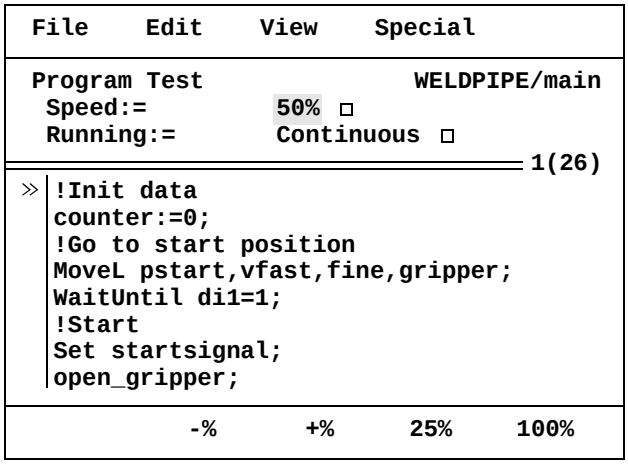


Figure 20 The speed can be changed (0 - 100%).

• Increase or decrease the speed by pressing the function keys ***-%*** or ***+%***. Correction then takes place in steps of 5%.
Set the speed to 25% or 100% by pressing the function key ***25%*** or ***100%***.

6.4 Choosing the execution mode

The program can be run in three different modes;

- continuous
- cycle (one cycle at a time)
- step-by-step (forwards or backwards, one instruction at a time).

Note The execution mode is automatically changed when switching between automatic and manual mode. The default set-up can be defined in the system parameters.

Choose continuous or cyclic running as follows:

• Select the upper part of the window by pressing the List key (if it is not already selected).

• Select the field ***Running***.

• Choose the program execution mode using the function key ***Cont*** or ***Cycle***.

Use the key to select the lower part of the window.

Use the function key ***Start*** to start program execution in the mode that you chose above. To step the program forwards/backwards, use the function keys ***FWD*** and ***BWD*** (see Figure 21).

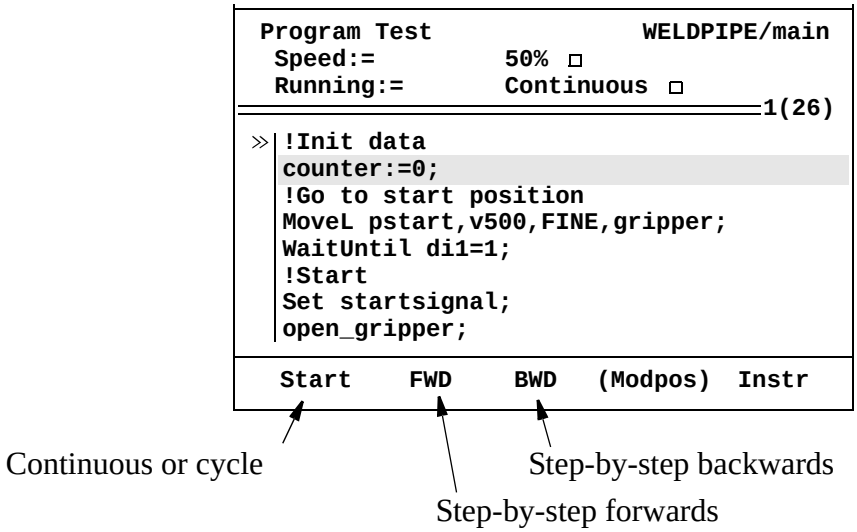


Figure 21 A program can be run in several different program execution modes.

Instructions act differently during step-by-step execution than during continuous execution. The principal differences are as follows:

- Positioning instructions are executed in the normal way, but the robot gets into position even when a fly-by point is programmed.
- Other instructions execute in the normal way when executing forwards and are skipped when executing backwards.

6.5 Starting program execution

• Choose the speed correction as above.

• Select the lower part of the window by pressing the List key (if it is not already selected).

When you start program execution, the robot will start to move. Peripheral equipment may also be started.
Make sure that everything is ready for program execution to begin and that nobody is in the safeguarded area around the robot. Starting the program incorrectly can injure someone, or damage the robot or other equipment.

• Set the robot into MOTORS ON mode by pressing the enabling device.

• Press the function key ***Start*** for continuous or cycle execution mode.
If you want to execute step-by-step, press the function key ***FWD*** or ***BWD*** instead.

When “Hold-to-run” is active, the following is applicable:

- press the ***Start*** key, release it, and press the Hold-to-run key. Keep this key depressed while the program is running, otherwise the program will stop (see Figure 22).

The start key should only be pressed once after each MOTORS ON, the Hold-to-run key can then be used to start and stop program execution.

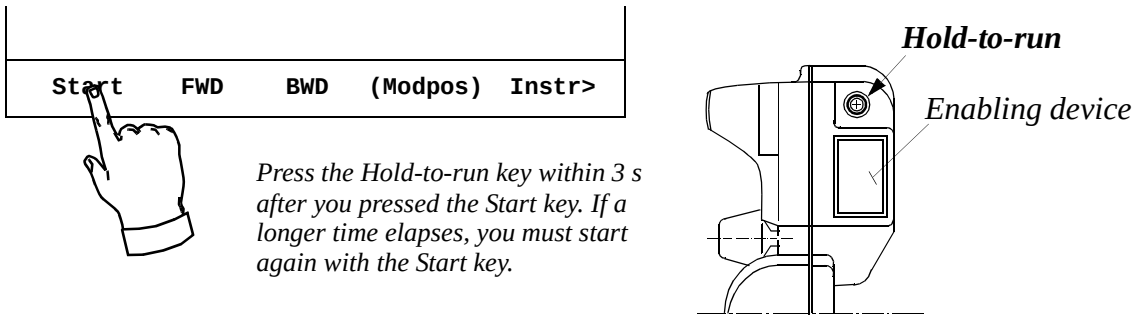

Start FWD BWD (Modpos) Instr>
Press the Hold-to-run key within 3 s after you pressed the Start key. If a longer time elapses, you must start again with the Start key.
Hold-to-run
Enabling device

Figure 22 The Hold-to-run key is located on the side of the teach pendant.

6.6 Stopping program execution

When Hold-to-run control is enabled

• Release the Hold-to-run key.

When Hold-to-run control is NOT enabled

• Press the Stop key on the teach pendant.

If the program execution mode is changed from continuous to step-by-step or cycle, the robot will stop automatically after it has completed the instruction or the cycle.

6.7 Where will the program start?

How do you recognise the program pointer?

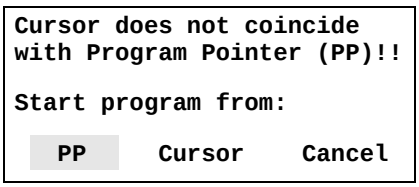
The program pointer shows how far the program has run and is marked with » in front of the instruction.

An instruction that has been fully executed is marked with an ×, but is only shown during instruction-by-instruction execution. If the cursor is positioned at this instruction, the program starts from the program pointer ». See the example below. (In all other cases, the cursor will define the instruction that will be executed when you press ***Start***.)

Example:

```
IF reg1=5 THEN
× reg2:=5;          The last instruction executed.
ELSE
   reg2:=8;
ENDIF
» Set do1           The next instruction to be executed
```

If the cursor is not located on the last instruction executed, then when you press ***Start***, an alert box will be displayed (because the program flow has been changed).
Select whether you wish to start from the program pointer (PP) or the cursor using the arrow keys:



• Press Enter ↵.

To move the cursor to the program pointer

• Choose **Special: Move cursor to PP.**

To move the program pointer to the cursor

• Choose **Special: Move PP to cursor.**

Note If the program pointer is moved into a FOR statement the program will run the rest of the FOR statement to the end, and then continue with the next statement.

To start the program from the beginning

• Choose **Special: Move PP to Main.**

The program pointer and the cursor are set to the first instruction in the main routine.

To start the program from a routine

The program pointer and cursor can be moved to any routine (procedure) with no parameters. If it is moved, the call hierarchy at that time will no longer be valid, which means that program execution continues from the start of the routine after the routine has been fully executed.

• Choose **Special: Move PP to Routine.**

A dialog box appears, displaying all possible routines.

• Select the desired routine and press ***OK***.

To execute a routine without losing the call hierarchy

A routine without parameters can be executed without losing the call hierarchy and program settings, e.g. program displacement, mechanical unit activation etc.

• Choose **Special: Call Routine**

A dialog box appears, displaying all possible routines.

• Select the desired routine and press ***OK***.

When the program pointer reaches the end of a called routine, you are asked whether to run the routine again or to return to the original program pointer where the “Call Routine” was performed.

To execute a service routine without losing the call hierarchy

A preconfigured service routine without parameters can be executed without losing the call hierarchy and program settings, e.g. program displacement, mechanical unit activation etc.

• Choose **Special: Call Service Routine**

A dialog box appears, displaying all possible service routines.

• Select the desired service routine and press ***OK***.

When the program pointer reaches the end of a called service routine, you are asked whether to run the service routine again or to return to the original program pointer where the “Call Service Routine” was performed.

To go to a position without moving the program pointer

Place the cursor on the position argument in the instruction. You can also select a position (robtarget) in the *Program Data* window.

• Choose **Special: Go to selected position.**

A dialog box appears, see Figure 23.

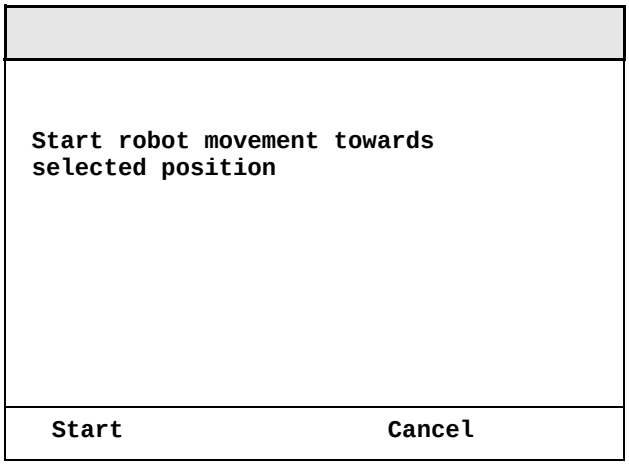


Figure 23 The Go to selected position dialog box.

• Press the function key ***Start*** to start the movement.

6.8 Simulating wait conditions

When the robot is stationary in a wait instruction, e.g. *WaitDI di1* or *WaitTime 3*, a dialog box is automatically displayed.

• To continue in the program without fulfilling the condition or time, press Enter ↵.

The dialog box will disappear automatically when the condition has been fulfilled.

7 Saving and Printing Programs

7.1 Saving the program on diskette or some other type of mass memory

To save a program that has been stored previously

• Choose **File: Save Program**.

The program is duplicated to mass memory and replaces the version that was last saved. If the file name or module name is not the same, the dialog **Save Program As** will be displayed automatically.

To save under a new name

• Choose **File: Save Program As**.

A dialog box appears, displaying all programs in the current directory (see Figure 24).

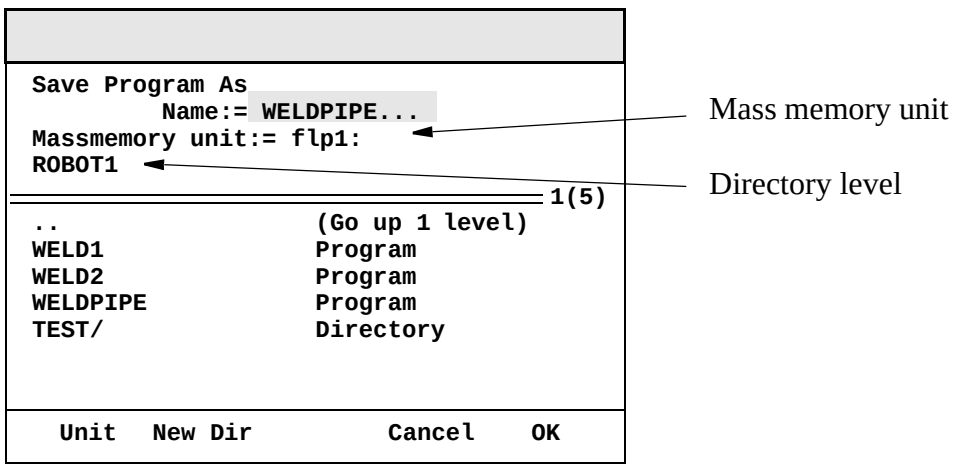


Figure 24 The dialog box used to store programs.

• If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

If the program is to be saved in another directory:

- Select the lower part of the window by pressing the List key.
- Choose the directory in which the program is to be saved. Move up or down in the directory by choosing either '. .' (up), or the desired directory (down) and press Enter ↵.
- Select the upper part of the window by pressing the List key.

- Press Enter ↵ when the field ***Name*** is selected.
- Specify the new name in the dialog box that appears. When you have finished entering text, press ***OK***.
- Choose ***OK*** to confirm the save.

Note If a file with the same name already exists, a warning will be given and you can choose to finish or continue.

Note If you have made a change in a system module, you will be requested to save this alteration.

7.2 Printing a program from the robot

Print a whole program

- Save the program on a diskette or the ramdisk, and print out from the File Manager. See Chapter 13 in this manual - *File Manager.*

Print a module

- Choose **File: Print**. The current module will be printed directly or saved to a file.

To be able to print, a printer must be connected to the robot controller.

7.3 Printing a program using a PC

A program can be printed using a personal computer. Most word-processing programs can be used, the only requirement being that the PC can support diskettes in DOS format.

- Store the program on a diskette.
- Load the program into the PC.
- Print the program.

If you do not wish to print out the position values of a position instruction, save the program using the command **File: Print** in the *Program* window and choose **Save to file**. Only the current module will be saved.

8 Changing the Program

Programs can be protected against alteration by making the appropriate settings in the system parameters. A password must then be used to make any changes. See chapter 12, System parameters *Topic: Teach Pendant*.

8.1 Selecting an instruction or an argument

A complete instruction or a single argument can be selected before a command is given to change the program. If you wish to change a single argument, it is often easiest to select the argument first. If you wish to change a complete instruction, you select the complete instruction. Often, e.g. when adding a totally new instruction, it does not make any difference whether the complete instruction, or an individual argument, is selected.

To select a complete instruction

Movement	Choose
Up one instruction	ArrowUp
Down one instruction	ArrowDown
To first instruction	**Edit: Goto Top**
To last instruction	**Edit: Goto Bottom**
To next page	NextPage
To previous page	PreviousPage

If the cursor is moved to the first line in a compound instruction (IF, FOR, WHILE or TEST), all instructions, including the last line (e.g. ENDIF), will be selected. If ArrowDown is then pressed, the instructions in the compound instruction will be selected, one after the other. Terminators (e.g. ENDIF, ELSE) cannot, however, be selected separately.

When the cursor is moved upwards to a compound instruction, ArrowUp can be used to select the instructions within that instruction, and ArrowLeft can be used to select the complete compound instruction.

To select a number of instructions

You can select a group of instructions that are in sequence.

- Select the first or the last instruction in the group.
- Choose **Edit: Mark**.
- Select the other instructions using ArrowUp or ArrowDown.

The selection will automatically be deactivated when **Edit: Cut** or **Edit: Copy** is chosen. You can also make the selection inactive by choosing **Edit: Unmark**.

To select an argument

- Use ArrowRight to move the cursor one argument to the right, or ArrowLeft to move the cursor one argument to the left.

8.2 Modifying the position in a positioning instruction

- Move the robot to the desired position.
- Select the instruction that is to be changed. For instructions containing more than one position, e.g. *MoveC*, select the position argument to be changed.
- Press the function key ***ModPos*** or choose **Edit: ModPos**.

Note An answer must be given to the password check and confirmation dialog if they have been set to active in the configuration. The default set-up is no password but with confirmation.

The maximum movement/reorientation of a position can be limited in the system parameters. If this has already been implemented, the system parameters must be changed in order to allow any greater changes of position. See chapter 12, System parameters *Topic: Teach Pendant*.

The old position is replaced by the current position of the robot. This position is related to the current tool and work object.

Note If a named position data is modified, all other instructions which refer to that position data will also be changed.

8.3 Tuning position during program execution

The tuning command makes it possible to tune the x, y and z coordinates of a robtarget during program execution. The function is valid only for named robtargets of the datatypes constant and persistent. It is not valid for positions represented by ‘*’ and robtargets of the datatype variable. The change is also valid in stopped mode.

- Start with the *Program Test* window.
- Press ***Start***.

The window *Program Run Info* appears (see Figure 25).

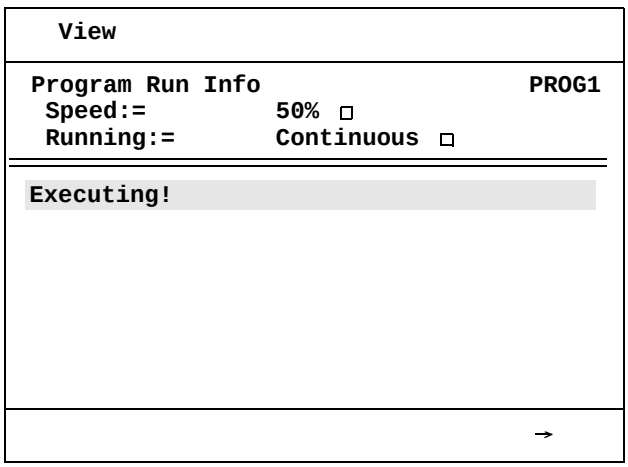


Figure 25 The window Program Run Info.

- Select **View: Position**.

The window *Program Run Position* appears (see Figure 26).

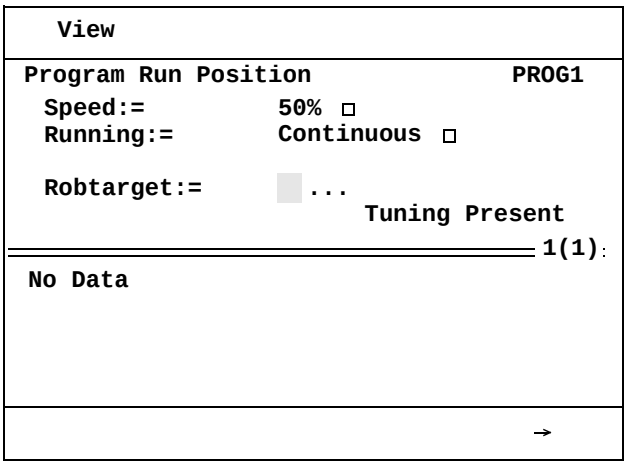


Figure 26 Window for tuning position during execution.

- Select the field ***Robtarget*** and press Enter ↵.
- Select the robtarget to be tuned. Only robtargets declared in the module chosen in the robtarget dialog are shown. To select a robtarget in an other module press ***Module***, select module, press Enter ↵ and continue selecting robtarget.
- Choose ***OK*** or press Enter ↵ to confirm the choice.

The x, y and z values of the chosen position are displayed (see Figure 27).

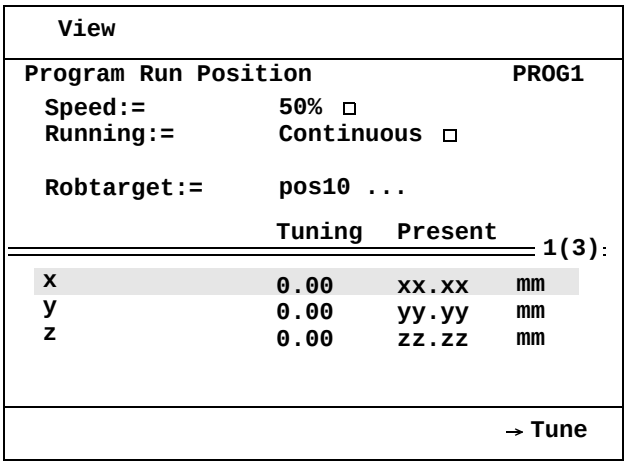


Figure 27 The Program Run Position window with a robtarget selected.

- Choose the x, y or z coordinate in the list.
- Press ***Tune***.

A dialog box will appear where you can tune the position.

- Enter the desired tuning value and press Enter ↵.
  - No change = 0
  - Maximum change in one step = ±10 mm.

Several steps can be entered. The position data is changed immediately after each step

but will not affect the robot path until the next instruction using this position data is executed. The values in the ***Present*** column will be used in this instruction.

The total tuning will be displayed in the ***Tuning*** column.

Note If a named position data is modified, all instructions which refer to that position data will be affected. Unnamed positions (marked as * in the instruction) cannot be tuned.

8.4 Changing an argument

- Select the argument that is to be changed.
- Press Enter ↵.

The dialog box used to program instruction arguments appears and the selected argument is marked with ”?” in front of it (see Figure 28).

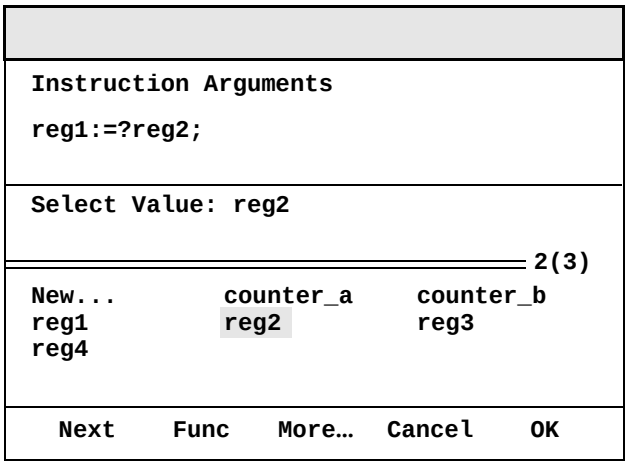


Figure 28 The dialog box used to change arguments. In this example, the argument reg2 will be changed.

The argument can now be changed in four different ways:

- by changing a numeric value; this alternative is used when a numeric value is to be specified, e.g. *5*, or when an argument is to be changed, e.g. from *reg2* to *reg3*;
  - Select the middle part of the dialog box and alternately do one of the following:
    - move the cursor to the left or right using ArrowLeft ◁ or ArrowRight ▷;
    - delete the character in front of the cursor by pressing Delete ⌫;
    - enter digits at the cursor using the numeric keyboard.
- by choosing data in the lower part of the of the dialog box; this alternative is used when the argument is to constitute a reference to data, e.g. *reg2;*
- by choosing a function; press the function key ***Func*** and select the desired alternative from the list. This alternative is used when an argument is to constitute a function call, e.g. *Offs(p1,5,0,0);*

A new dialog box that can be used to program function arguments appears. Use the function key ***Skip*** to delete optional arguments that are not to be included.

- by entering an optional expression, press the function key ***More***; this alternative is used when the argument is to constitute an expression with several components, e.g. *reg1+reg2* or *reg1>5*, or a string value, e.g. *“Producing part A”*.

  For more information, see *Expressions* on page 18.

• If desired, choose ***Next*** to change the next argument.

• Choose ***OK*** to confirm.

Note You can also use ***Copy*** and ***Paste*** to change arguments.

Note Any changes in an active position instruction (except for ***ModPos***) will be valid first for the next execution of the instruction. To get an immediate result, choose **Special: Move PP to cursor.**

8.5 Adding optional arguments

Optional arguments of an instruction are not normally included when programming an instruction, but have to be added afterwards.

• Select the instruction that is to be modified.

• Press the function key ***OptArg***.
If you are in the *Program Test* window, you must first select the whole instruction, then press Enter ↵ and then ***OptArg***.

A dialog box appears, displaying all arguments that the current instruction can possibly have. The arguments not included in the instruction are enclosed within square brackets (see Figure 29).

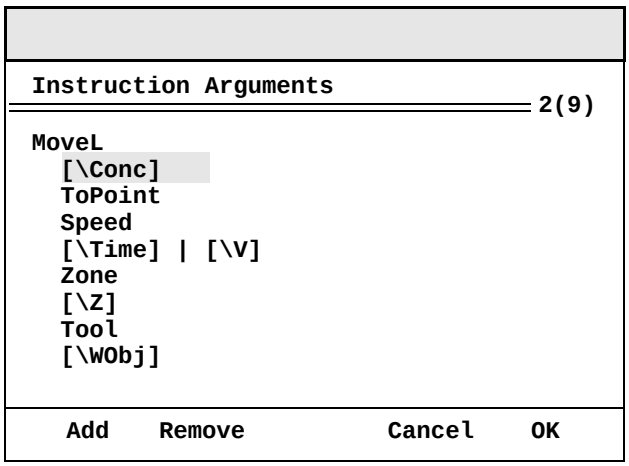


Figure 29 The dialog box used to add optional arguments.

• Add an optional argument by selecting the desired argument and pressing ***Add***.

Some arguments (displayed on the same line) cannot exist simultaneously in an instruction. When such an argument is added, the corresponding mutually exclusive argument is automatically removed.

An optional argument can also be removed by selecting the desired argument and pressing the function key ***Remove***.

• Choose ***OK*** to confirm the change.

8.6 Changing the structure of an IF, FOR or TEST instruction

• Select the complete instruction that is to be changed.

• Press Enter ↵.

A dialog box appears, displaying the structure that the instruction can have. Structure parts not included in the instruction are enclosed within square brackets (see Figure 30).

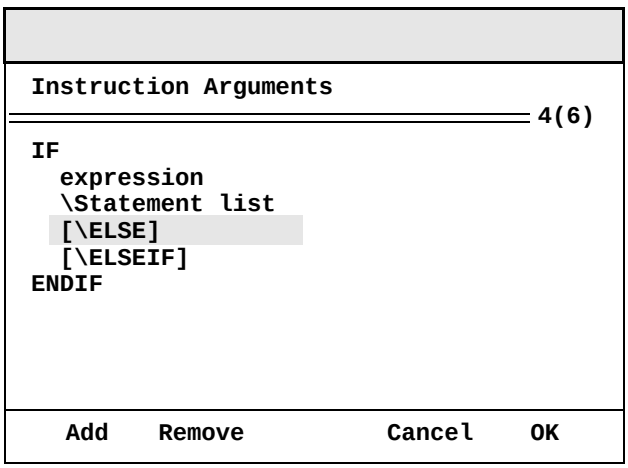


Figure 30 The dialog box used to change the structure of an IF instruction.

• Add part of the structure by selecting the desired part and pressing ***Add***.

• Remove a part of the structure by selecting the desired part and pressing ***Remove***.

• Choose ***OK*** to confirm the change.

Note If you want to add more than one ELSEIF or CASE, these can be added in the Program window using ***Copy*** and ***Paste***. Different CASE statements, such as CASE 1, 2, 3, can also be added using ***Copy*** and ***Paste***.

8.7 Changing the name or declaration of a routine

• Choose **View: Routine**.

• Select the desired routine.

• Press the function key ***Decl***.

A dialog box appears, displaying the routine declaration.

• Make whatever changes you wish to make (see *Creating a new routine* on page 10).

• Choose ***OK*** to confirm the change(s).

8.8 Deleting an instruction or an argument

• Select the instruction or the argument you wish to delete.

• Press Delete ⌫ .

Note An answer must be given to the password check and confirmation dialog if they have been set to active in the configuration. The default set-up is no password but with confirmation.

If an argument is compulsory (required), it will be replaced by <...>.

8.9 Deleting a routine

• Choose **View: Routines**.

• Select the desired routine.

• Press Delete ⌫ .

Note An answer must be given to the password check and confirmation dialog if they have been set to active in the configuration. The default set-up is no password but with confirmation.

8.10 Undo latest action

• Choose **Edit: Undo.**

The command **Undo** performs an undo operation on the latest performed action in the window selected. Undo is enabled in the *Program Instr*, *Test*, *Data* and *Routine* windows. The *Program Instr* and *Program Test* windows share the same undo buffer. The *Data* window and *Routine* window each have their own undo buffer. The undo command line tells you which command is to be undone (see Figure 31). If there is nothing to be undone or if the undo buffer has been lost, the undo command is disabled. This is shown as **(Undo)** on the menu command line.

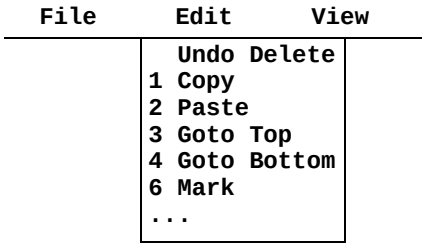


Figure 31 In this example the latest Delete command can be undone.

Operations that are possible to undo:

- **Delete, Cut, Paste** are always possible to undo.
- **Change Selected** is used to change the arguments to an instruction. When the operation is ordered either via **Edit: Change Selected** or by pressing Enter ↵, a copy of the entire selected instruction is made. If the operation is to be undone, the instruction is replaced by this copy.
- **Optional Arguments** is used to add or remove arguments to instructions or procedure calls. Undo is handled as Change Selected.
- **New Instruction** is undone by removing the latest inserted instruction.

Each of these operations clears previous undo buffers. E.g. it will not be possible to Undo a previous delete operation when a new instruction has been inserted.

When Undo is performed, the part of the program that is affected will be shown.

Limitations: Operations that change data values, e.g. ModPos and **Edit:Value**, cannot be undone.

9 Special Editing Functions

9.1 Search & replace

The search and replace function makes it possible to search for and replace data names in the program. It is also possible to search for/replace procedure/function calls. Instruction names can also be changed, e.g. from *MoveL* to *MoveJ*.

Choose **Edit: Search** in the *Program Test* or *Program Instr* window.

A dialog box appears (see Figure 32).

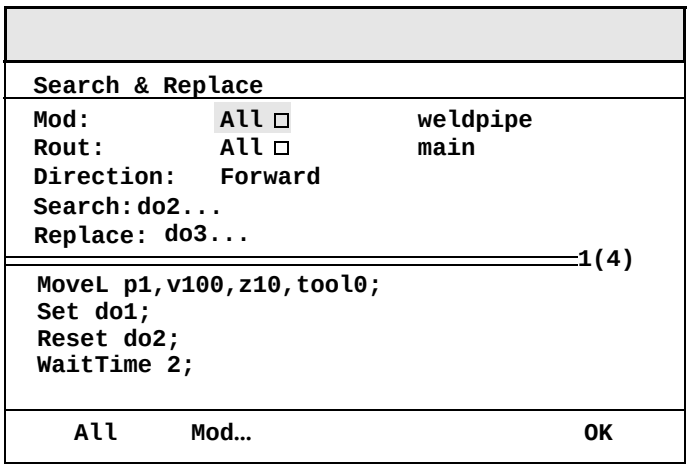


Figure 32 The search and replace dialog.

• Define how the search is to be carried out by defining the following fields.

Field	Description
Mod	If “All modules” is selected, the current module name is shown to the right of the field. If ***Mod*** or Enter ↵ are pressed, a list of available modules is shown, and a specific module can be selected.
Rout	Searching/replacing is performed in all available routines in the module selected from the module field. If ***Rout*** or Enter ↵ are pressed, a list of available routines is shown, and a specific routine can be selected.
Direction	Directions for searching.
Search	Selection field where a list of available names to search for is shown when you press Enter ↵.
Replace	Selection field where a list of available names to replace is shown when you press Enter ↵.

To start a search

• Move the cursor to the lower part of the window using the list key (see Figure 33).

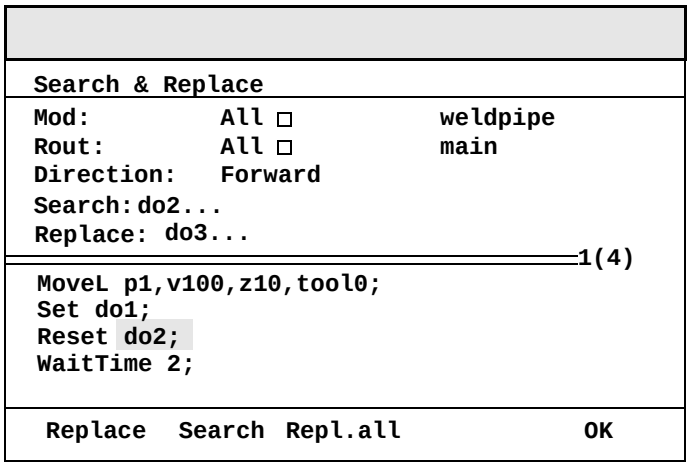


Figure 33 Search & Replace dialog when the program is selected.

• Press the function key ***Search*** to start the search.

The first match will be selected in the lower part of the window.

• Press ***Replace*** to replace the selected text or press ***Repl.all*** to replace all matches without having to confirm.

• Continue the search with ***Search***.

• Press ***OK*** to end the search.

9.2 Mirroring

The mirror function can be applied to any routine in a program.

Mirroring a routine means that a copy of the routine is created with all positions mirrored in a specific mirror plane.

The new, mirrored routine will be given a new name (a default name is proposed). All stored data of type robtarget, used in the routine, will be mirrored and stored with a new name (the old name ending with _m). All immediate robtarget data, shown with an “*”, in movement instructions will also be mirrored.

What does mirrored mean?

In general, all data of the type robtarget, used in the routine, will be mirrored. It makes no difference whether the robtarget data is declared as a constant (which it should be), as a persistent or as an ordinary variable. Any other data, e.g. of type pos, pose, orient, etc., will not be mirrored. Mirroring data only affects the initialization value, i.e. any current value will be ignored. This means that if a robtarget variable has been defined without an init value, this variable will not be mirrored.

The mirroring works as follows:

- The new routine is scanned for any local robtarget data, declared inside the routine with an init value. All such data’s init values are mirrored.
- Then the new routine is scanned for any statement with one or more arguments of type robtarget.
- When such a statement is found, the following actions will take place:
  - If the argument is programmed with a reference to a local variable or a constant, this argument will be ignored, since it has already been mirrored as described above.
  - If the argument is programmed with an immediate robtarget data, shown with an asterisk “*”, then this value will be mirrored directly.
  - If the argument is programmed with a reference to a global variable, persistent or a constant, defined outside the routine with an init value, then a duplicate is created and stored in the module with a new name (the old name ending with _m). The init value of this new data is mirrored, and after that the argument in the statement is changed to the new name. This means that the module data list will expand with a number of new mirrored robtarget data.

Error handlers or backward handlers, if any, in the routine, are not mirrored.

Mirror plane

The mirror function will mirror all positions, mentioned above, in the mirror plane, i.e. the mirrored position will be located symmetrically on the other side of the plane, relative to the original position. The mirror plane is always the xy-plane of an object frame, used for mirroring. This object frame is defined by a work object data, e.g. with the name MIRROR_FRAME. The work object MIRROR_FRAME uses, as all work objects, two frames for defining the object frame: the user frame and object frame.

The object frame is defined relative to the user frame, and the user frame is defined relative to the world frame. Usually, the user frame is set equal to the unity frame and, in such a case, the object frame is defined relative to the world frame (see Figure 34).

The mirror frame must be stated in the mirror dialogue.

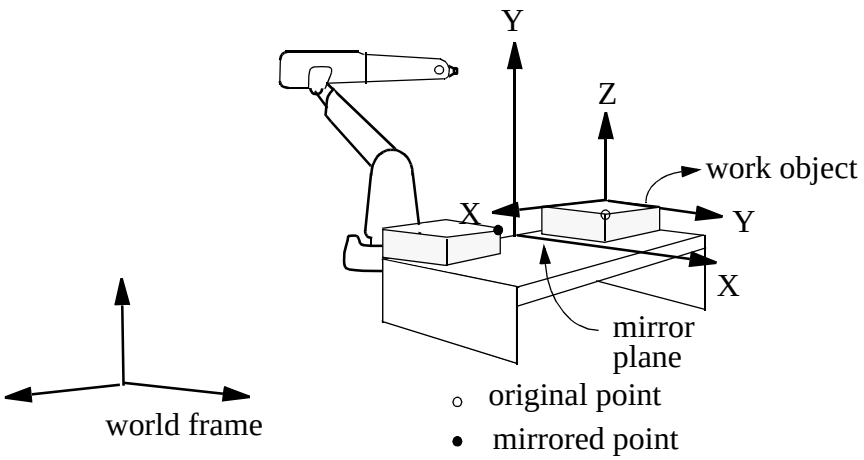


Figure 34 The mirror plane.

Work object

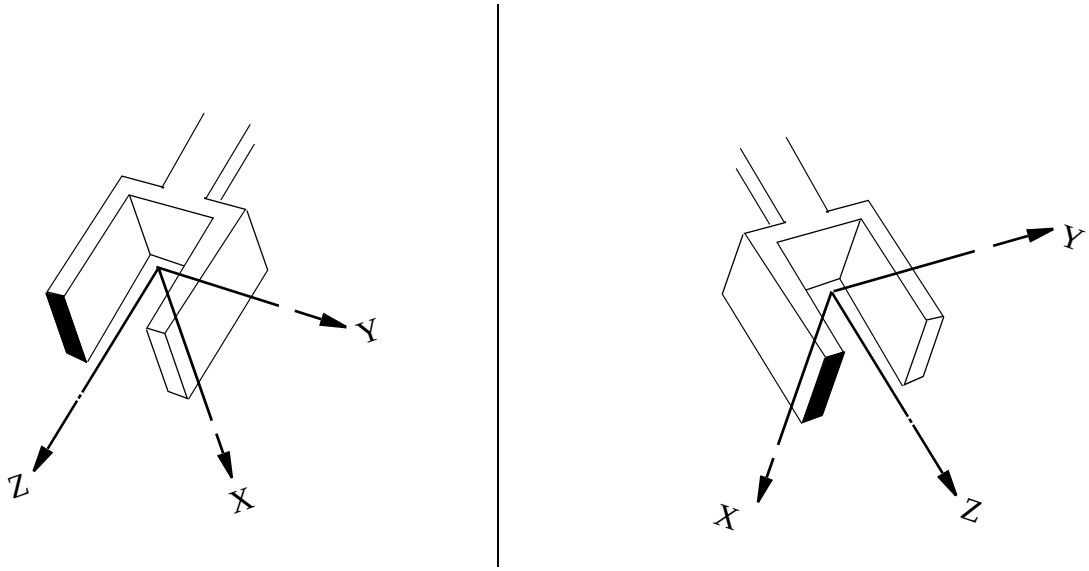
All positions which are to be mirrored are related to a specific work object frame. This means that the coordinates of the robtarget data are expressed relative to this work object frame (see the figure above). Furthermore, the mirrored position will be related to the same work object frame.

In the dialogue, before mirroring, this specific work object must be stated. This work object will be used as the reference frame for all variables that are to be mirrored.

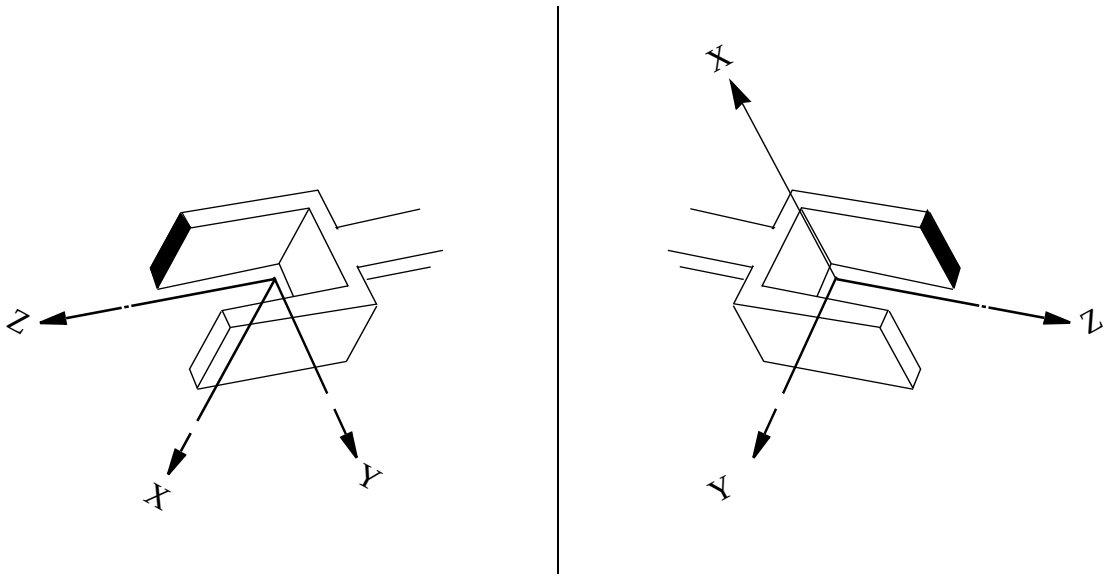
IMPORTANT:
Be sure to state the same work object as was originally used when defining the robtarget data, and which was used as a parameter in the movement instructions. If no work object was used, the wobj0 should be stated.

Mirroring of orientation

The orientation of the robtarget position is also mirrored. This mirroring of the orientation can be done in two different ways, where either the x and z axes are mirrored or the y and z axes (see Figure 35). The method used, x or y axis (the z axis is always mirrored), is dependent on the tool used and how the tool coordinate system is defined. In the mirror dialogue, the method must be stated.



Mirroring of x- and z-axes



Mirroring of y- and z-axes

Figure 35 Two different ways of mirroring.

Configuration

The configuration will not be mirrored, which means that, after mirroring, it has to be carefully checked by executing the path in test mode. If the configuration has to be changed, this must be done manually and the position corrected with a modpos command.

Mirror example 1, one robot

A programmed routine, org, is stored in the robot's memory. A mirrored copy of this routine is to be created and stored with the name mir in memory. All positions are related to the work object, wobj3. The mirror plane is known from three positions in the plane, p1, p2 and p3.

An original position in org, pos, is mirrored to pos_m (See Figure 36).

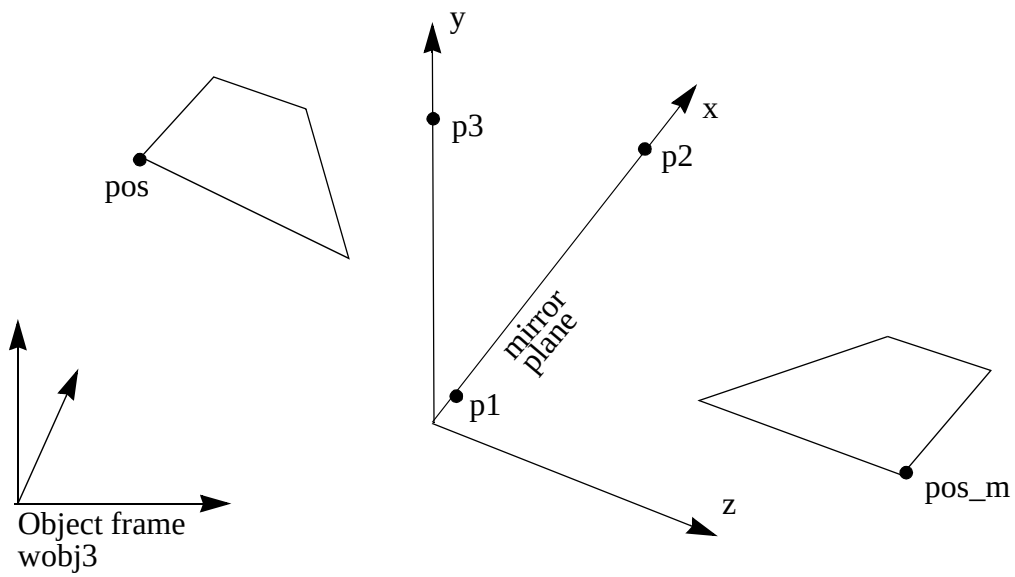


Figure 36 Mirroring of a routine, using one robot.

To perform this mirroring, the mirror frame must first be defined. To do this, start off by creating a new work object and name it mirror or whatever. Then, use the three points, p1 to p3, to define the object coordinate system with the help of the robot (see Chapter 10, Calibration).

After this, the routine, org, can be mirrored using wobj3 and mirror as input data.

Mirror example 2, two robots

In this case, a routine, org, created on one robot, is to be mirrored and used on another robot. Suppose that a spot welding robot, robot 1, is used for the left side of a car body. When the program for the left side is done, it should be mirrored and used again for the right side by robot 2.

The original program, org, is programmed relative to a work object, wobj1, which is defined with the help of three points, A, B and C on the left side of the car body, using the "3-point" method, (see Chapter 10, Calibration). The mirrored program, mir, is to be related to a corresponding work object, wobj1, defined by the corresponding points D, E and F on the right side of the car body. Wobj1 for robot2 is defined with robot2, using the same "3-point" method. Note that since the points D, E, F are reflected images of points A, B and C, the wobj1 for robot2 will also be mirrored. One of the consequences of this is that the z-axis will point downwards.

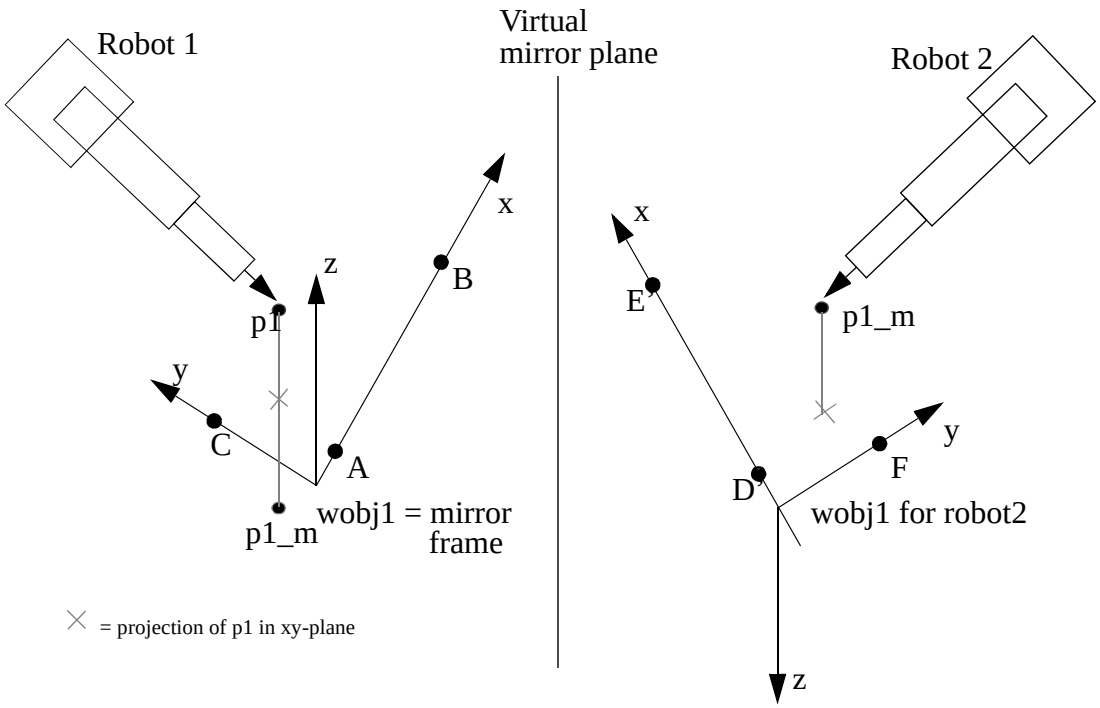


Figure 37 Mirroring of a routine, using two robots.

After the work object, wobj1, has been defined, all programming is done in this frame. Then the program is mirrored using the same wobj1 frame as the mirroring frame. A position, p1, will be mirrored to the new position p1_m.

After this, the mirrored program is moved to robot 2, using the work object wobj1, as described above. This means that the mirrored position, p1_m, will be “turned up” as if it were mirrored in a “virtual” mirror plane between the two robots (see Figure 37).

Mirror function dialogue

• Choose **View: Routines**

• Select the routine to be mirrored.

• Choose **Special: Mirror**

A dialog box appears (see Figure 38).

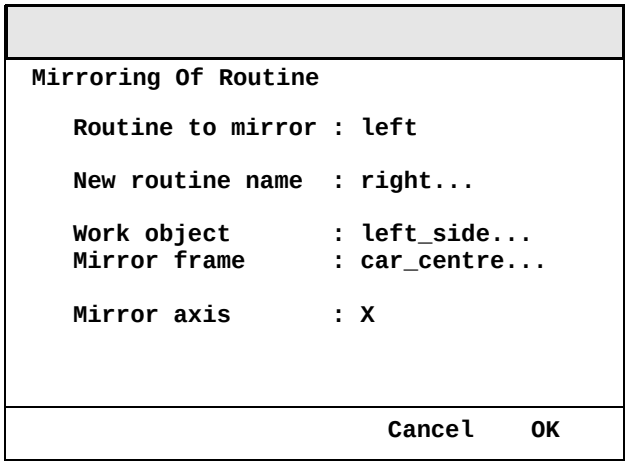


Figure 38 Mirror function dialogue.

• Define how the mirroring is to be performed in the fields below.

Field	Description
Routine to mirror	The name of the routine that will be mirrored.
New routine name	The mirrored routine will be given this name. If the Enter ↵ key is pressed when this field is selected, a text input dialog will be displayed.
Work object	The work object to be used when applying the mirror function on robtarget variables. If the Enter ↵ key is pressed, the work object selection dialogue will be displayed.
Mirror frame	The frame to be used as the mirror plane. The frame is of the type wobjdata. If the Enter ↵ key is pressed, a mirror frame selection dialogue will be displayed.
Mirror axis	Specifies the mirroring of orientation. When this field is selected, the function key bar shows the alternatives X and Y. The mirroring of orientation is then selected by pressing the corresponding function key.

• Start the mirroring with ***OK***.

10 Creating Data

10.1 What is data?

Data is used to store values that can be used at a later stage in the program. Data is grouped into different data types that describe its contents and its field of application.

Data type	Used for:
num	Numeric values (registers, counters)
bool	Logical values (true or false)
robtarget	Position data
tooldata	Tool data (see Chapter 10, Calibration)
wobjdata	Work objects (see Chapter 10, Calibration)
pose	Program displacement frames (see Chapter 10, Calibration)

For more detailed information on data and its contents, see the appropriate data type in RAPID Reference Manual - *Data Types*.

Data must be defined (declared) before it can be used. However, depending on the configuration of the robot, there is usually a number of predefined data.

Data can be defined as constants, variables or persistents:

- The value of a *constant* can only be changed manually.
- A *variable* can also be changed by the program, but its initialisation value is automatically set when:
  - the program is read from diskette or the like,
  - the program is started from the beginning, i.e. from the first instruction in the main routine,
  - the program pointer is moved to the beginning of a routine by choosing **Test: Move PP To Routine**, or to the beginning of a program by choosing **Test: Move PP To Main**.
- A *persistent* can be described as a variable whose initialisation value is constantly updated so that it corresponds to the current value. Thus, its value is not changed when the program is started from the beginning. If the program is output to a diskette, the new initialisation value is stored.

10.2 The *Program Data* window (used to manage data)

• Choose **View: Data** to open the Program Data window.

The window displays all data of the type last selected. The current values are also displayed (see Figure 39).

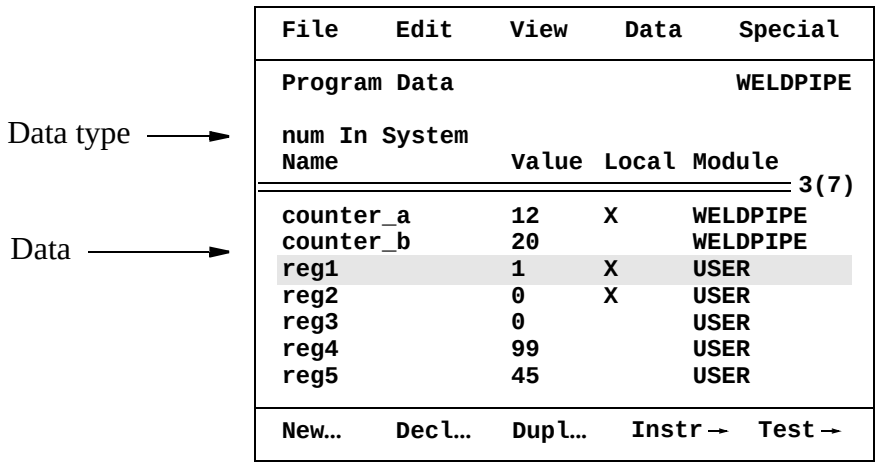


Figure 39 All data of a given type are displayed in the Program Data window.

To choose a new data type in the Program Data window

• Open the window *Program Data Types* by choosing **View: Data Types**.

The *Program Data Types* window opens and displays all data types that have at least one declared data (see Figure 40).

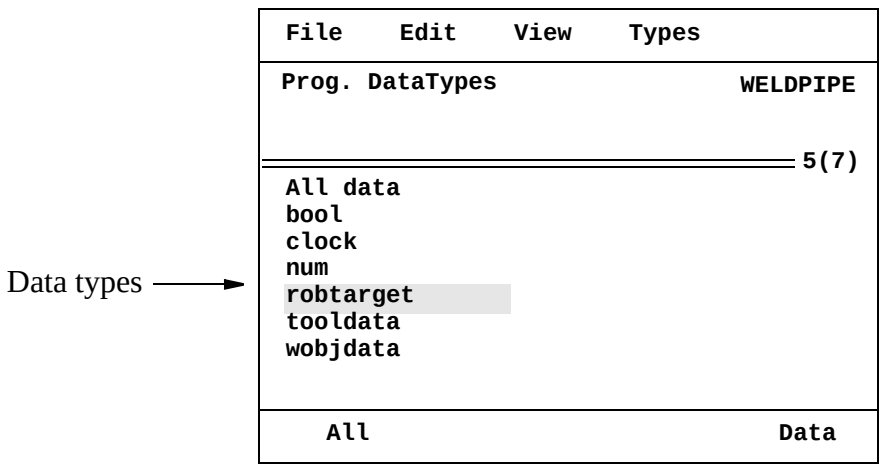


Figure 40 The Program DataTypes window is used to change the data type.

• Select the desired data type and press Enter ↵. If the desired type is not displayed in the window, you can call up all data types by pressing ***All type*** or choosing **Types: All Types**.

All data can be chosen by selecting ***All data***.

Data for a selected type can be chosen by pressing ***Data*** or Enter ↵.

10.3 Creating new data

• Open the *Program Data* window by choosing **View: Data ...**

The *Program Data* window is opened and displays all data of the type last selected.

If you wish to create data of a type other than that displayed, choose **View: Data Types**, select the desired data type and press Enter ↵.

• Press the function key ***New.***

A dialog box appears, displaying the name of the data (see Figure 41). The name of the data is set to *xxxN* where *xxx* describes the data type and *N* is a number incremented each time this type of data is created. The first data of the type clock is named *clock1*, the second, *clock2*, etc. Some data types are abbreviated, e.g.:

Data type	Predefined name	Data type	Predefined name
num	reg*N*	loaddata	load*N*
robtarget	p*N*	tooldata	tool*N*
bool	flag*N*	speeddata	speed*N*

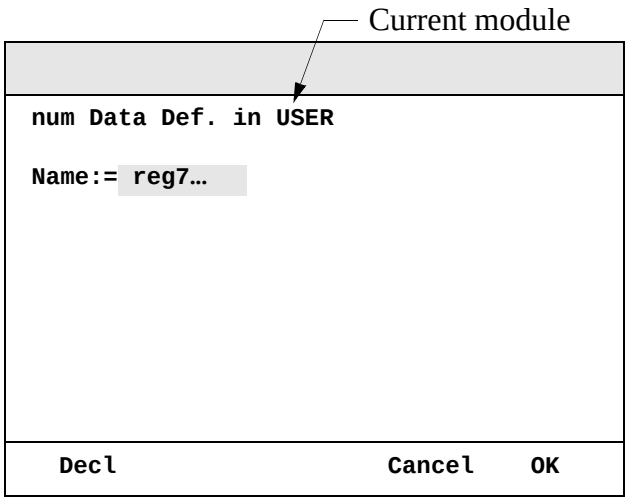


Figure 41 New data is created.

• Change the name by pressing Enter ↵ and specify a new name.

The data will automatically be given characteristics that are best suited to the current type, but these can be changed when necessary.

Normally, data is stored as a part of the program. However, when data is to be present in the memory, irrespective of which program is loaded, it is stored in the system module *User*. Examples of this type of data are:

- tools and work objects; changing this data will affect all programs.
- registers and other data that are not to be initialised when a program changes.

When you wish to save in the current module and with standard characteristics, you can finish by pressing ***OK***. In other cases the characteristic must be defined.

• Press the function key ***Decl***.

A dialog box appears, displaying the basic data declaration (see Figure 42).

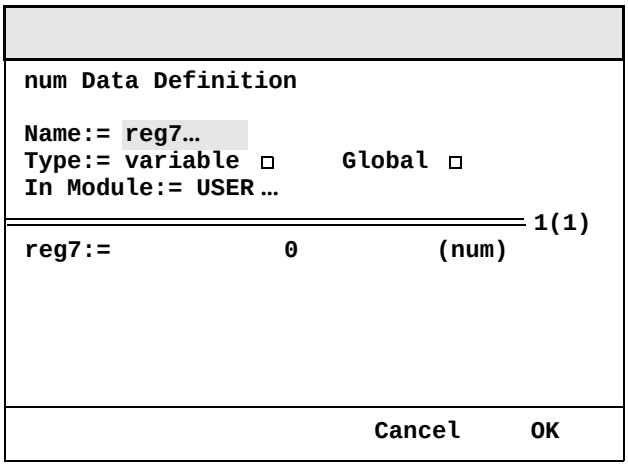


Figure 42 A data declaration includes the name and characteristics of the data.

• Select the appropriate field and specify the desired characteristics by:

- pressing Enter [↵] and specifying the desired alternative in the dialog box that appears (fields marked with ...)
- choosing an alternative using the function keys (fields marked with ▫)
- specifying the value directly using the numeric keyboard (numeric initial value).

Field	Description
Name	The name of the data (a maximum of 16 characters).
Type	Specifies whether the data is to be a constant (***Const***), variable (***Var***) or persistent variable (***Pers***).
Global/Local	Specifies the scope attribute for the data. Default for the datatype is set in **File:Preferences**. See *Default data Global/ Local* on page 61.
In Module	The module in which the new data will be used.
Initial value	A value assigned to the data when, e.g. reading from a diskette. Change the value by pressing [▤] and enter the new initial value.

• Choose ***OK*** to approve the definition.

Tip It is sometimes easier to create new data by duplicating and changing existing data.

10.4 Creating new array data

• Open the *Program Data* window by choosing **View: Data.**

The *Program Data* window is opened and displays all data of the type last selected.

If you wish to create data of a type other than that displayed, choose **View: Data Types**, select the desired data type and press Enter [↵].

• Select **Data:New Array**

A dialog box appears, asking for the number of dimensions, 1, 2 or 3. Make a choice and press Enter ↵.

A dialog box appears, displaying the basic array data declaration (see Figure 43).

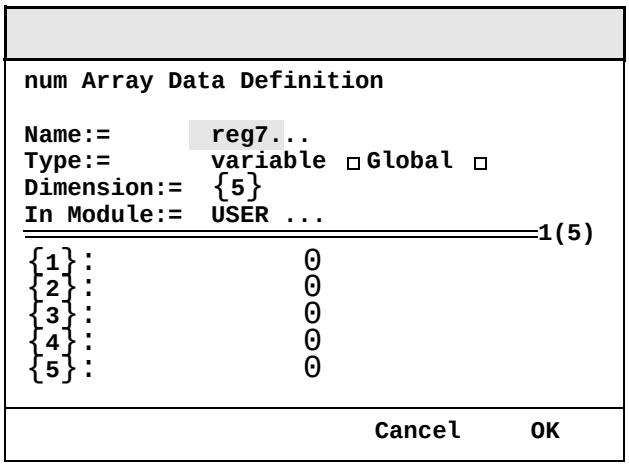


Figure 43 An array data declaration includes the name and characteristics of the data.

• Select the appropriate field and specify the desired characteristics by:

- pressing Enter ↵ and specifying the desired alternative in the dialog box that appears (fields marked with ...)
- choosing an alternative using the function keys (fields marked with □)
- specifying the value directly using the numeric keyboard (numeric initial value).

Field	Description
Name	The name of the data (a maximum of 16 characters).
Type	Specifies whether the data is to be a constant (***Const***), variable (***Var***) or persistent variable (***Pers***).
Global/Local	Specifies the scope attribute for the data. Default for the datatype is set in **File:Preferences**. See *Default data Global/ Local* on page 61.
Dimension	Size of the chosen dimensions.
In Module	The module in which the new data will be stored.
Initial value	A value assigned to the data when, e.g. reading from a diskette. Change the value by pressing and enter the new initial value.

• Choose ***OK*** to approve the definition or ***Cancel*** to abort the definition.

10.5 Duplicating data

- Open the window *Program Data* by choosing **View: Data**.
- Select the data to be duplicated.
- Press the function key ***Dupl***.
- Specify the new name in the dialog box that appears.
- Choose ***OK*** to confirm the duplication.

10.6 Storing position data using the robot

- Open the *Jogging* window and specify the tool and work object on which the position is to be based.
- Jog the robot to the desired position.
- Create new data as described in *Creating Data* on page 45. Specify the data type *robtarget*.

The current position of the robot will be automatically stored as an initial value.

10.7 Routine data

Normally, data – *program data* – can be accessed from anywhere in the program. Data can also be linked to a specific routine – *routine data* – and, in this case, exists locally within the routine.

- Open the *Program Data* window by choosing **View: Data**.
- Choose **Data: In Routine ...**

The window will then display the routine data for the current routine. The window is identical to the window shown in Figure 39, except that it displays the routine name after the program name.

Now you can create and change routine data in the same way as for program data.

11 Changing Data

11.1 Viewing and possibly changing the current value

- Select the desired data in an instruction.
- Choose **Edit: Value**.

A dialog box will appear, displaying the current value (see the example in Figure 44). For more detailed information on the meaning of the various components, see the appropriate data type in the RAPID Reference Manual.

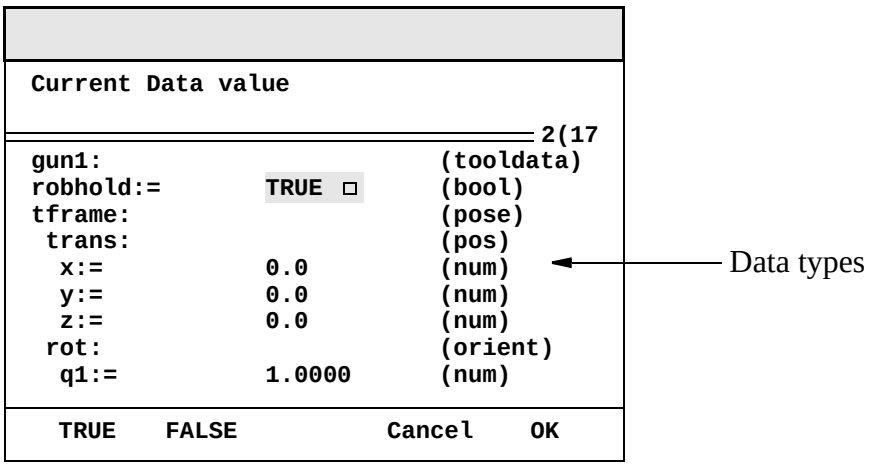


Figure 44 The dialog box used to change values.

- Change the value by selecting the desired field, then:
  - Choose an alternative using the function keys.
  - Specify the value directly using the numeric keyboard.
- Choose ***OK*** to confirm the change.

You can also open the dialog box as follows:

- Choose **View: Data**.
- Select the desired data. If you wish to view data of a type other than that displayed, choose **Data: Datatypes** and select the desired data type.
- Press Enter ↵ or choose **Data: Value**.

Continue as above when the dialog box appears.

11.2 Changing data names or declarations

- Choose **View: Data**.
- Select the desired data. If you wish to view data of a type other than that displayed, choose **Data: Datatypes** and select the desired data type.
- Press the function key ***Decl***.

A dialog box appears, displaying the data declaration.

- Change the name and declaration as described in *Creating Data* on page 45.
- Choose **OK** to confirm the change.

11.3 Deleting data

- Choose **View: Data**.
- Select the desired data.
- Press Delete ⌫.
- Press ***OK*** to confirm the deletion.

12 Error Handling

Each routine has an error handler that can be programmed to deal with any errors that occur during program execution. In this way, some errors (listed below) can be dealt with automatically by the program:

- when no search stop is obtained during a search,
- when a file cannot be opened,
- when there is division by 0.
- Other errors are listed under *Data Types - errnum - Predefined Data* (RAPID Reference Manual).

The error handler is programmed in the normal way using RAPID instructions. When an error occurs, a jump is made to the error handler in the routine in which the error occurred.

If there is no error handler, a jump is made instead to the error handler in the routine that called the routine in question. A general error handler for the whole program can therefore be created in the main routine. If there is no error handler when an error occurs, program execution will stop and an error message will be displayed.

The error can then be remedied in the error handler and the program can be automatically restarted as in the example in Figure 45.

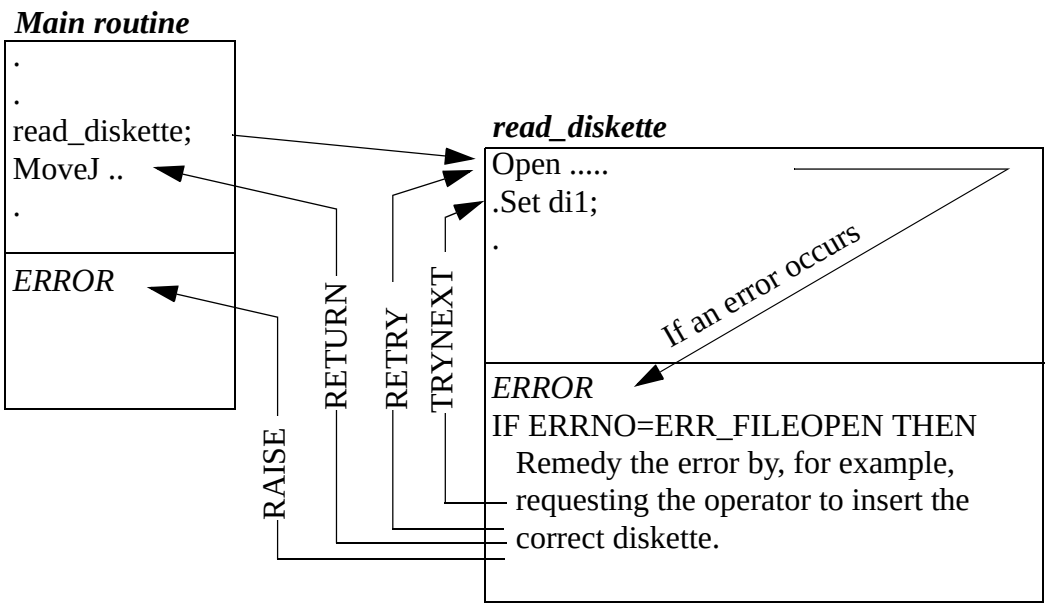


Figure 45 The program can be restarted from the error handler in various ways.

If the program cannot read a diskette, a jump is made to the error handler of the routine, where the error is remedied. The program can then be restarted by re-executing (RETRY) the instruction that caused the error, executing the next instruction (TRYNEXT) or by returning (RETURN) to the calling routine. The error can also be remedied in the error handler of the main routine (RAISE).

To create an error handler

- Choose **View: Routines**.
- Choose the routine to which the error handler is to belong.
- Choose **Routine: Add Error Handler**.

To program the error handler

- Choose the routine to which the error handler is to belong.
- In the *Routine* window: Choose **Routine: Error Handler.**
  In other windows: Choose **View: Error Handler**.
- Program the error handler in the usual way.
- Return to the main part of the routine by choosing **View: Instr**.

To remove an error handler

- Choose **View: Routines**.
- Choose the routine to which the error handler is to belong.
- Choose **Routine: Remove Error Handler**.

13 Using Modules

13.1 What is a module?

The robot program can be subdivided into *program modules*, each module containing a group of routines and data. In addition to this, system modules which are always present in the memory, can be used (see Figure 46).

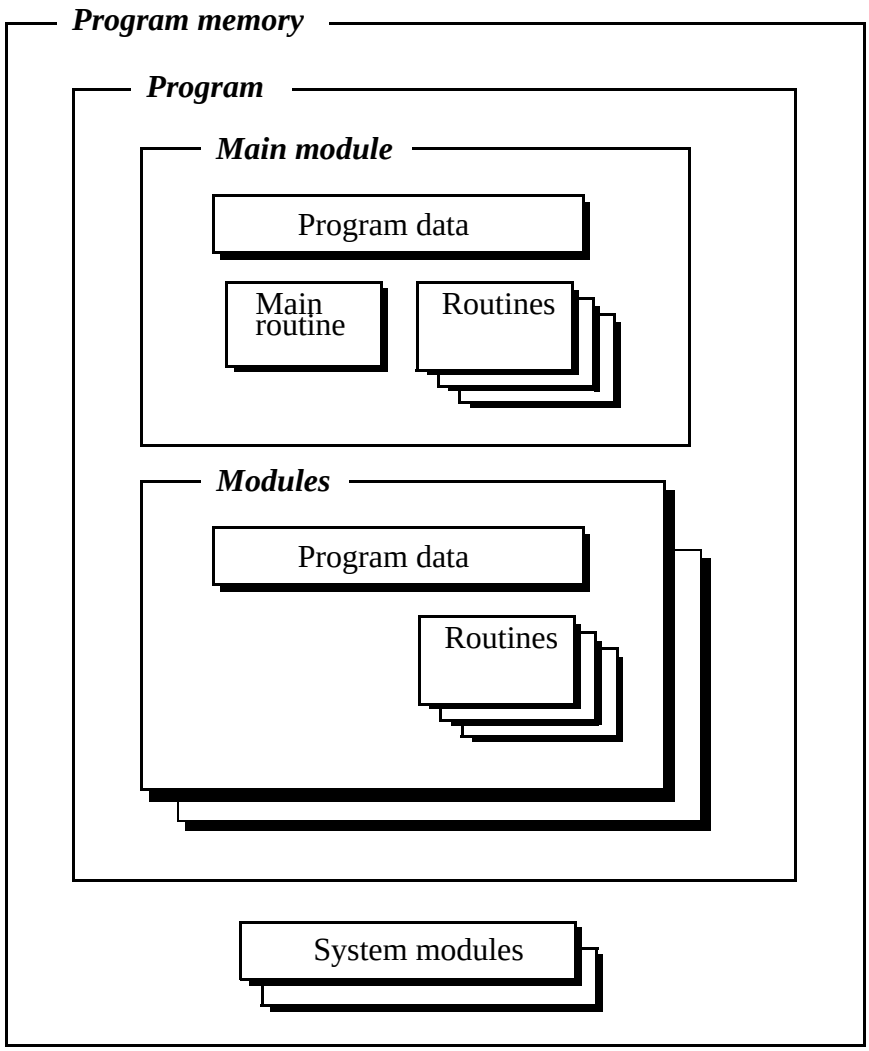


Figure 46 Routines and data can be grouped together to form modules.

The entire program or separate modules can be stored on diskette or some other type of mass memory. System modules are automatically loaded when the system is cold-started.

A module can include, for example:

- general routines for many different installations,
- positions generated via CAD,
- routines for a certain type of external equipment, such as a workpiece manipulator.

System modules can, for example, include general data (e.g. tool data) for all programs used in the same robot.

The main routine of the program is located in one of the modules (the module with the same name as the program).

Both program and system modules work in the same way once they have been loaded into the memory. All modules can normally be edited using the teach pendant, but, as system modules are often write-protected, the write protection must first be removed.

13.2 Choosing modules

• Choose **View: Modules**.

The window *Program Modules* displays all modules present in the program memory (see Figure 47).

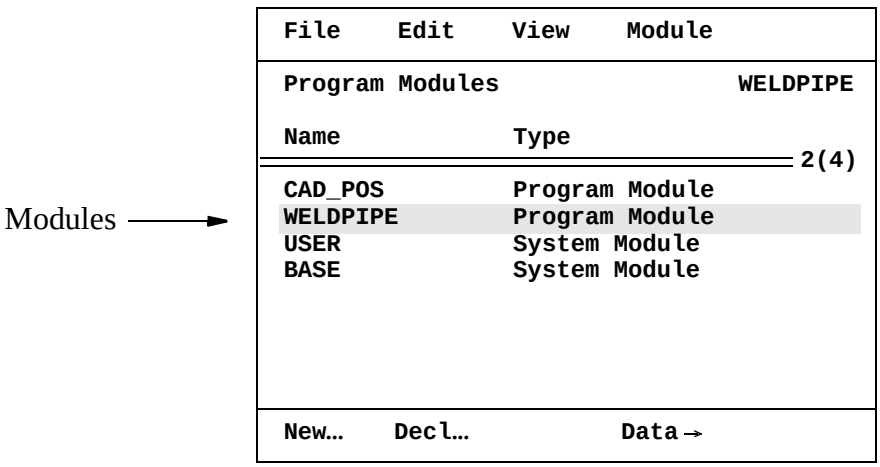


Figure 47 The Program Modules window displays all modules in the task program.

• Select the desired module.

• Press Enter ↵.

The *Program Routines* window, in which you can choose the desired routine, opens.

13.3 Creating a new module

• Open the window *Program Modules* by choosing **View: Modules**.

• Press the function key ***New***.

A dialog box appears, displaying the basic module declaration (see Figure 48). The name of the routine is set to *moduleN*, where *N* is a number incremented each time a routine is created.

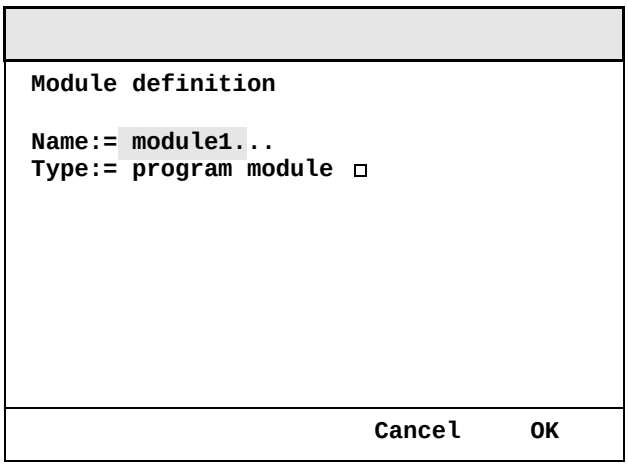


Figure 48 A module declaration specifies the name and characteristics of a module.

• Change the name and characteristics of the module by selecting the appropriate field, then:

- Press Enter ↵ and specify the desired alternative in the dialog box that appears on the display (fields marked with...).
- Choose an alternative using the function keys (fields marked with □).

Field	Description
Name	The name of the module (a maximum of 16 characters).
Type	Specify whether the module is to be a program or system module.

• Press ***OK*** to end the module declaration.

13.4 Changing the name or declaration of a module

• Choose **View: Module**.

• Select the desired module.

• Press the function key ***Decl***.

A dialog box appears, displaying the module declaration.

• Make whatever changes you wish to make (see *Creating a new module* on page 56).

• Choose ***OK*** to confirm the change(s).

13.5 Reading a program module from diskette or some other type of mass memory

• Choose **File: Open**.

A dialog box appears, displaying all modules and programs in the current directory (see Figure 49).

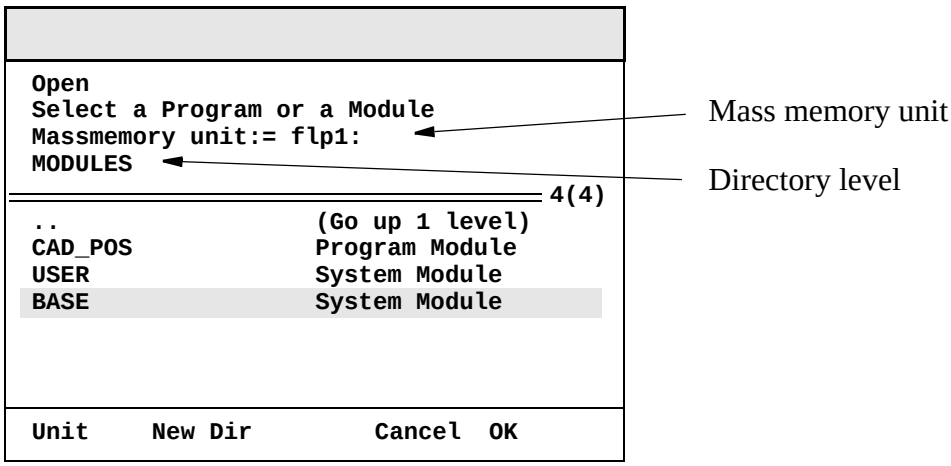


Figure 49 The dialog box used to read modules.

• If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

• Select the desired program module. Move up or down in the directory by choosing either '. .' (up), or the desired directory (down) and press Enter ↵.

• Choose ***OK*** to confirm.

The specified module will then be read to the robot memory and added to the rest of the program.

13.6 Deleting program modules from the program

• Open the window *Program Modules* by choosing **View: Modules**.

• Select the desired module.

• Press Delete ⌫.

Note An answer must be given to the password check and confirmation dialog if they have been set to active in the configuration. The default set-up is no password but with confirmation.

13.7 Listing all routines in all modules

Usually only the routines contained in the current module are displayed in the *Program Routines* window. You can, however, change this so that all routines in all modules are displayed.

- Open the window *Program Routines* by choosing **View: Routines**.
- Choose **Routine: In System**.

To list only the routines in the current module again, choose **Routine: In Module**.

13.8 Duplicating a routine from one module to another

- Choose the module in which the new routine is to be included.
- List all routines by choosing **Routine: In System** in the *Program Routines* window.
- Select the routine to be duplicated.
- Continue in the normal way, as described in *Duplicating a routine* on page 11.

13.9 Listing all data in the current module

Usually the data contained in all modules is displayed in the *Program Data* window. You can, however, change this to display only the data in the current module.

- Open the window *Program Data* by choosing **View: Data**.
- Choose **Data: In Module**.

To list all program data in all modules again, choose **Data: In System**.

13.10 Duplicating data from one module to another

Data can be duplicated from one module to another. Routine data cannot, however, be duplicated.

- Choose the module in which the new data is to be included.
- Select the data to be duplicated in the *Program Data* window.
- Continue in the normal way, as described in *Duplicating data* on page 50.

13.11 Saving modules on diskette or some other type of mass memory

To save a module that has been stored previously

- Open the window *Program Modules* by choosing **View: Modules**.
- Select the module to be saved.
- Choose **File: Save Module**.

The module is duplicated to mass memory and replaces the last version saved.

To save under a new name

• Open the window *Program Modules* by choosing **View: Modules**.

• Select the module to be saved.

• Choose **File: Save Module As**.

A dialog box appears, displaying all modules and programs in the current directory (see Figure 50).

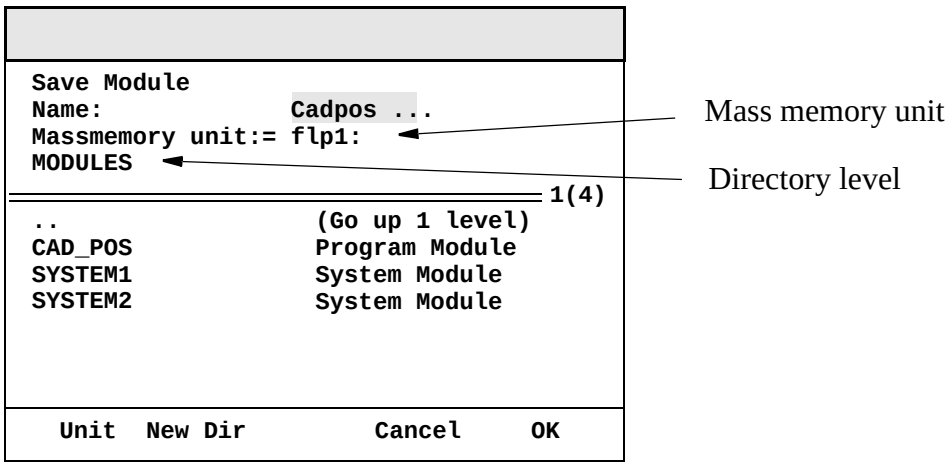


Figure 50 The dialog box used to store modules.

• If necessary, change the mass memory unit by pressing ***Unit*** until the correct unit is displayed.

• Choose the directory in which the module is to be saved. Move up or down in the directory by choosing either '. .' (up), or the desired directory (down) and press Enter ↵. Create a new directory by pressing ***New Dir***.

• Press Enter ↵ when the field ***Name*** is selected.

• Specify the new name (using the numeric keyboard) in the dialog box that appears. Press ***OK*** when you have finished entering the new name.

• Choose ***OK*** to confirm the save.

13.12 Calling up the complete module list

• Choose **View: Modules**.

• Select the desired module.

• Choose **Module: Module List**.

The complete module is displayed, including its data declarations and routines. It cannot, however, be changed.

• Exit the module list by pressing ***OK***.

14 Preferences

14.1 Defining the *Most Common* instruction pick list

You can define the contents of the *Most Common* instruction pick list to obtain a pick list of the instructions you use most.

- Choose **File: Preferences**.
- Select, for example, ***Most Common SetUp 1***.
- Press Enter ↵.

All instructions and procedures are displayed. Those included in the pick list are marked with an *x* to the left of their names (see Figure 51).

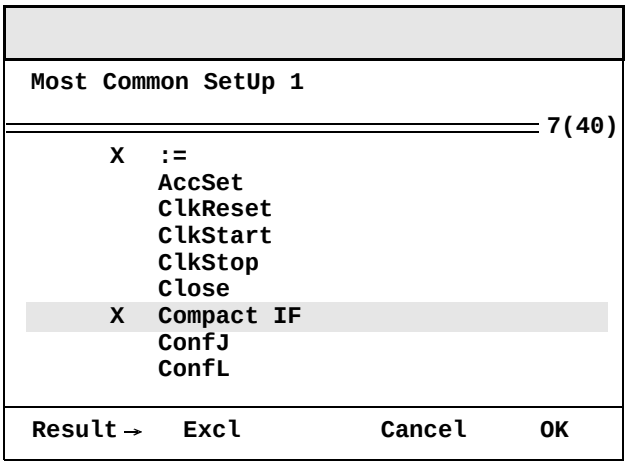


Figure 51 You specify the instructions to be included in the list in the Most Common Setup dialog box.

- Add an instruction by selecting the appropriate instruction and pressing ***Incl.*** That instruction will then be marked with an *x* to its left.
- Remove an instruction by selecting the appropriate instruction and pressing ***Excl.*** The instruction will still be displayed in the window but the *x* to its left will disappear.
- Press ***Result***.

The instructions included in the pick list are displayed (see Figure 52).

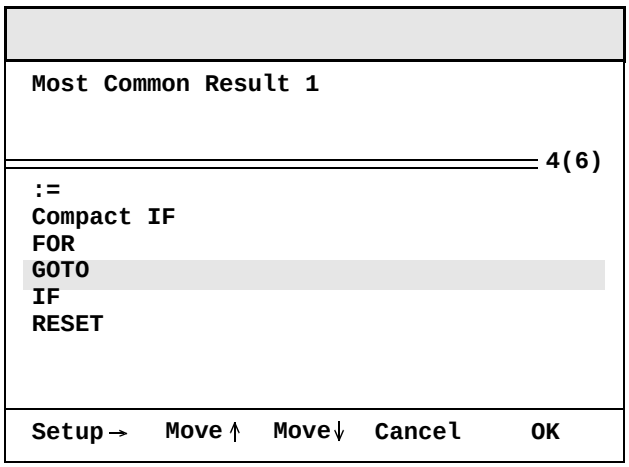


Figure 52 You specify the order of the instructions in the list in the Most Common Result dialog box.

- Change the order of the instructions using ***Move*↑** and ***Move*↓**. ***Move*↑** moves the selected instruction up one step and ***Move*↓** moves it down one step.
- When the definition is ready, press ***OK***.
  To return to the *Most Common Setup* dialog box, press ***Setup*** instead.

The current Most Common list is automatically chosen as the active pick list. The various Most Common lists can be chosen from the IPL2 menu in the Program Instr window.

Note This definition is stored in the system parameters (topic Teach Pendant) which should be saved from the *System Parameters* window.

14.2 Default data Global/Local

You can decide what scope new data of a specific datatype should have by default. The scope is either ***Global***, reachable from other modules, or ***Local***, only reachable in the module where the data is declared. When new data are created the setting in this list will be used for the data scope attribute.

- Choose **File: Preferences.**
- Select ***Default data Global/Local*** and press Enter [↵].
- A list with all available datatypes are presented. Mark the datatype you want to change and press ***Global*** or ***Local***.

Note This definition is stored in the system parameters (topic Teach Pendant) which should be saved from the *System Parameters* window.

For more information regarding Global and local data, see RAPID Reference Manual.

14.3 Defining programming rule for robot positions

It is possible to select whether or not to automatically create new robtargets when programming move instructions.

Automatic name sequence

When a *Move* instruction is programmed, a new robtarget is automatically created. If the last used ToPoint was named, i.e not a “*”, a new robtarget will be created and named according to a sequential naming rule. For example p10, p20, p30 or p12, p13, p14 etc.

Dialog with next robtarget selected

This rule is used when robtargets are created in advance. When a *Move* instruction is programmed **no** robtarget is created. Instead the instruction argument dialog is opened with the next sequential robtarget selected. For example, if the last used robtarget was *p100*, the instruction argument dialog will be opened with *p110* selected.

Dialog with * selected

Same as “Dialog with next robtarget selected” except that the instruction argument dialog is opened with the “*” selected.

• Choose **File: Preferences**.

• Select ***Robtarget programming rule***

• Select a programming rule and press **OK.**

CONTENTS

Page

1 Programming a Position 3
1.1 Positioning instructions 3
1.2 Programming an offset 6
2 Changing the Value of an Output 7
3 Waiting 8
3.1 Wating for an input 8
3.2 Waiting a specific amount of time 10
4 Controlling the Program Flow 10
4.1 Calling a subroutine 10
4.2 Program control within a routine 11
5 Assigning a Value to Data (Registers) 14

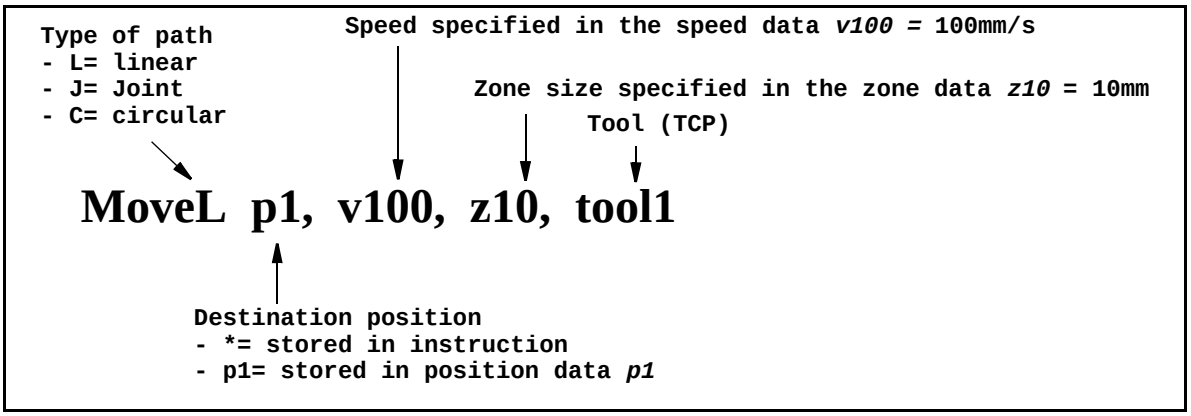
The programming language RAPID

1 Programming a Position

1.1 Positioning instructions

A positioning instruction contains the following information:

- Type of path (e.g. linear, joint motion)
- The destination position to which the robot is to move
- Speed
- Zone size (accuracy), i.e. how close the robot must be to the destination position before it can start to move towards the next position. If *fine* is chosen, the robot moves to the position.
- Current tool (TCP).


Type of path
- L= linear
- J= Joint
- C= circular
Speed specified in the speed data v100 = 100mm/s
Zone size specified in the zone data z10 = 10mm
Tool (TCP)
MoveL p1, v100, z10, tool1
Destination position
- *= stored in instruction
- p1= stored in position data p1

The speed and zone size refer to different data, which includes the desired speed in mm/s, zone size in mm, etc. You can create and name this data yourself, but, the most commonly used values are already available.

You specify the tool – its dimensions and weight – in the tool data (see Chapter 10, Calibration). The TCP of the tool is moved to the specified destination position when the instruction is executed (see Figure 1).

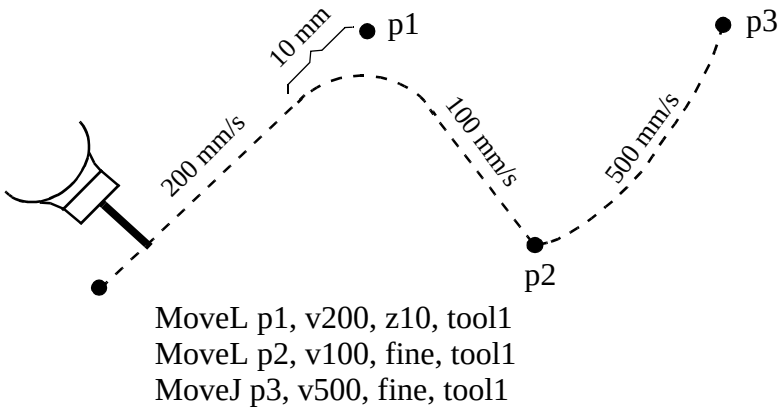


Figure 1 Positioning the robot.

Apart from these arguments, a positioning instruction may contain optional arguments, e.g. arguments used to specify the positioning time. See the appropriate instruction in RAPID Reference Manual for more details.

• Jog the robot to the desired destination position.

• Call up the instruction pick list by choosing **IPL1: Motion&Process**.

The program and specified pick list will then appear in the window (see Figure 2).

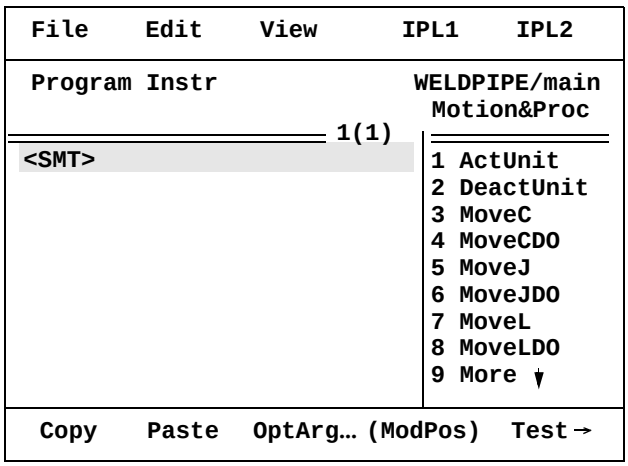


Figure 2 The dialog box used to program positioning instructions.

• Choose the desired instruction by pressing the appropriate numeric key.

The instruction will be added directly to the program, as illustrated in Figure 3. The arguments are set automatically.

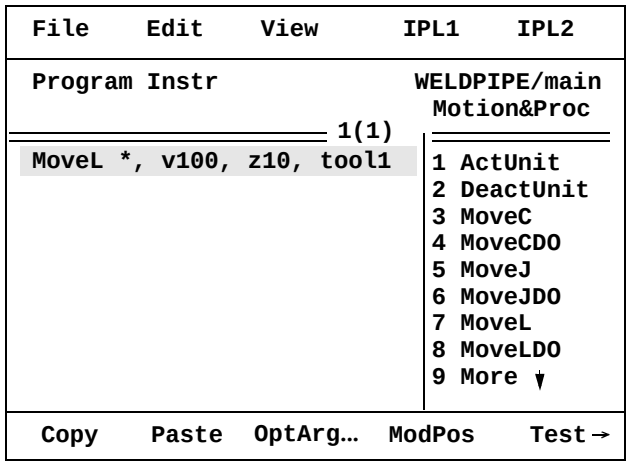


Figure 3 A positioning instruction is added directly to the program.

If the correct argument was chosen, the instruction is now ready for use. However, we will continue and change the speed and zone size.

• Select the argument you wish to change (*v100* in this example).

• Press Enter [↵].

The dialog box, used to program instruction arguments, appears. The selected argument is marked with a ? in front of it (see Figure 4). The lower part of the box displays all available speed data that can be selected.

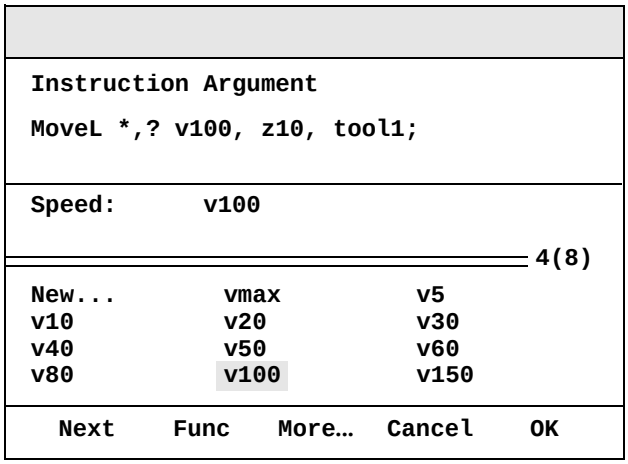


Figure 4 The dialog box used to change the speed.

• Select the desired speed.

• Go to the next argument (zone data) by pressing ***Next***.

All available zone data will be displayed (see Figure 5).

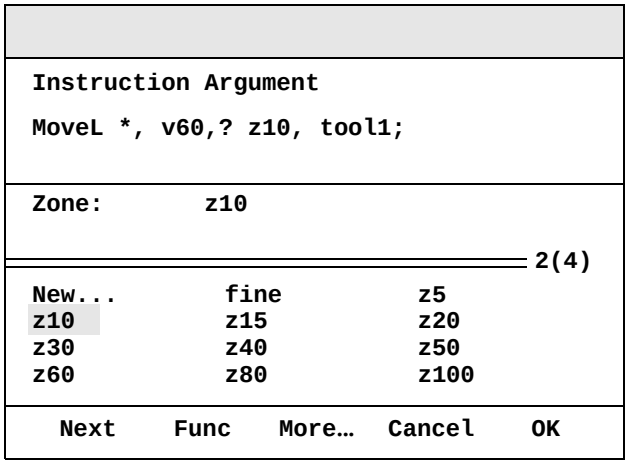


Figure 5 The dialog box used to change zone data.

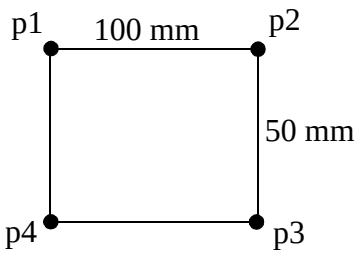
• Select the desired zone size.

• Choose ***OK*** to confirm the change.

The instruction is now ready for use.

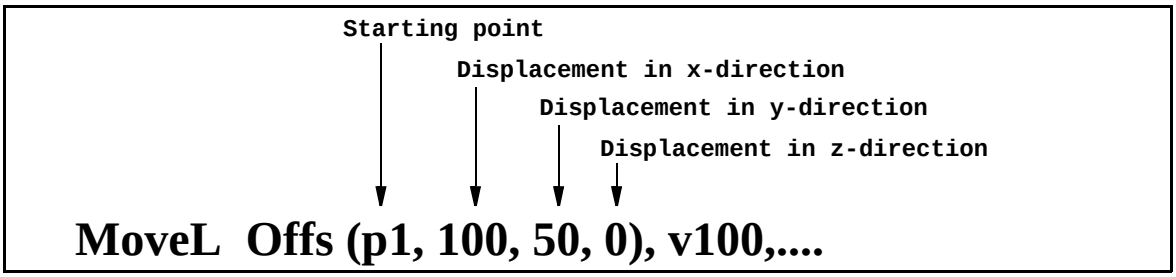
1.2 Programming an offset

Sometimes it is easier to define a position as an offset from a given position. If, for example, you know the exact dimensions of a work object, it will only be necessary to jog to one position (see Figure 6).



MoveL p1, ...	MoveL p1, ...
MoveL p2, ...	MoveL Offs (p1, 100, 0, 0), ...
MoveL p3, ...	MoveL Offs (p1, 100, 50, 0), ...
MoveL p4, ...	MoveL Offs (p1, 0, 50, 0), ...
MoveL p1, ...	MoveL p1, ...

Figure 6 Two different ways of programming a movement.



• Program a positioning instruction as described in *Programming a Position* on page 3.

• Select the position argument and press Enter ↵ .

• Press ***Func***.

• Select the function ***Offs*** and press Enter ↵ .

A dialog box appears in which you can enter the arguments of the function (See Figure 7).

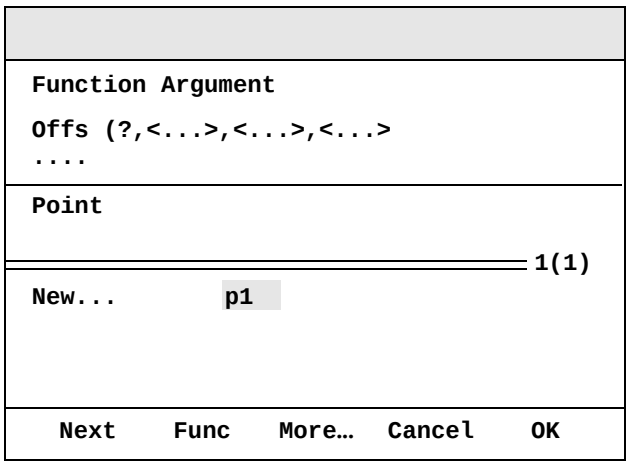


Figure 7 The dialog box used to set an offset.

• Select the starting point.

• Press ***Next***.

• Enter the offset (the offset value) in the x-direction using the numeric keyboard.

• Press ***Next***.

• Enter the offset in the y-direction using the numeric keyboard.

• Press ***Next***.

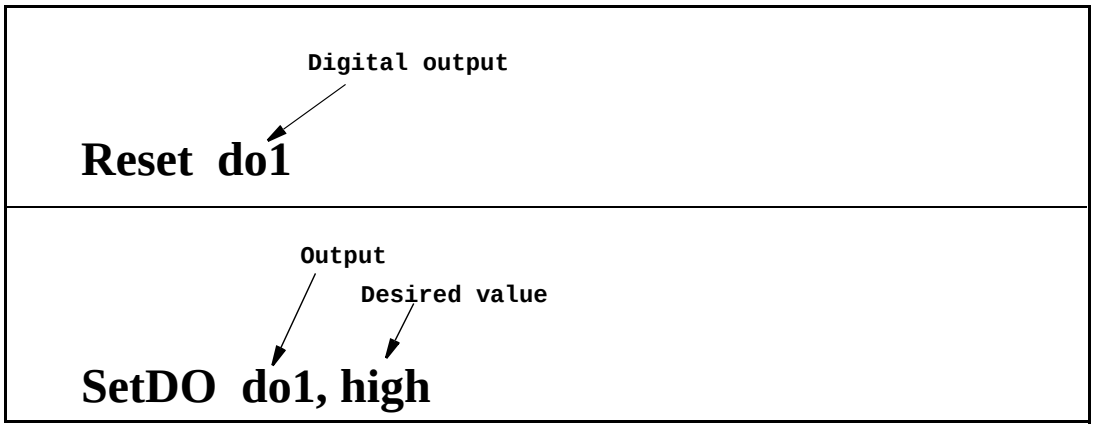
• Enter the offset in the z-direction using the numeric keyboard.

• Press ***OK***.

2 Changing the Value of an Output

An output instruction contains the following information:

- information on the output to be changed,
- information on the desired value.



• Call up the instruction pick list for I/O instructions by choosing **IPL1: IO**.

• Choose the desired instruction by pressing the appropriate numeric key.

You must now specify the output to be changed. All the different robot outputs are displayed for this purpose (see Figure 8).

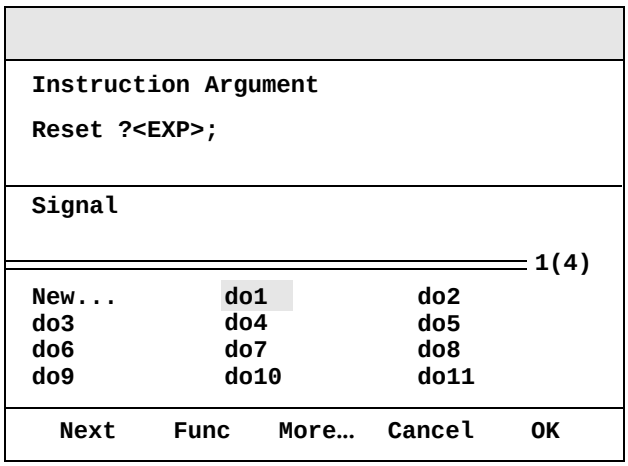


Figure 8 The dialog box used to define an output.

• Select the desired output.

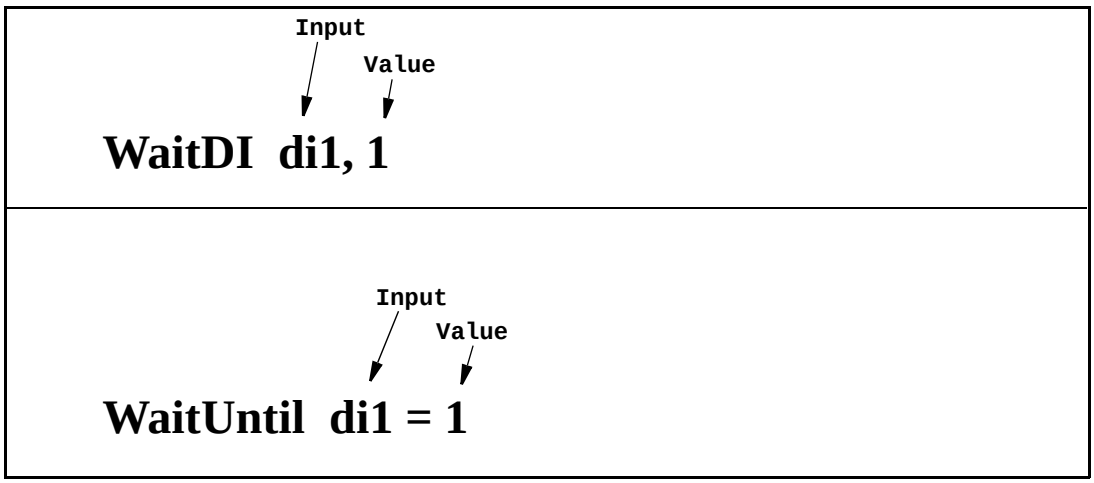
• Choose ***OK*** to confirm.

3 Waiting

3.1 Wating for an input

A wait-until-input instruction contains the following information:

- the name of the input,
- the input value necessary for program execution to continue.



The *WaitUntil* instruction can also be used to wait for several inputs.

• Choose **IPL1: Various**.

• Select the instruction *WaitDI*.

You must now specify the condition that must be satisfied before the program execution is to continue. You do this using the dialog box illustrated in Figure 9.

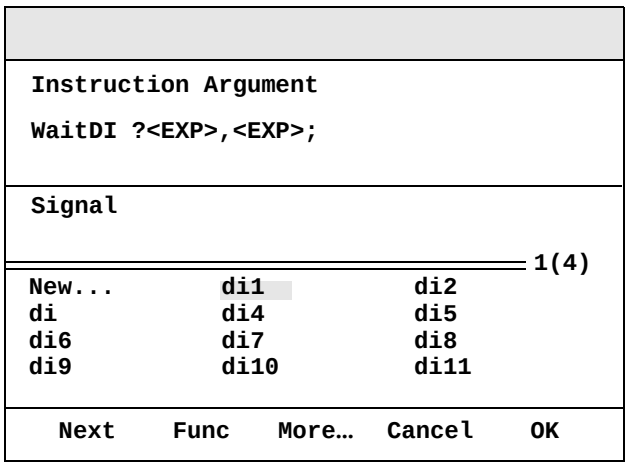


Figure 9 The dialog box used to define an input.

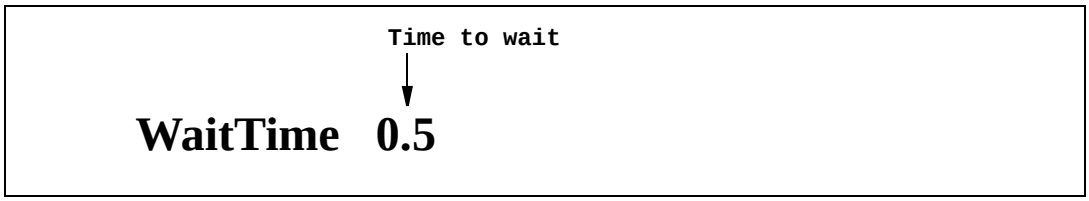
• Select the desired input.

• Choose ***Next*** to define the next argument, i.e. the value of the input.

• Enter the input value using the numeric keyboard.

• Press ***OK*** to confirm.

3.2 Waiting a specific amount of time



- Choose **IPL1:Various**.
- Select the instruction *WaitTime.*

A dialog box appears in which you can enter the time (see Figure 10).

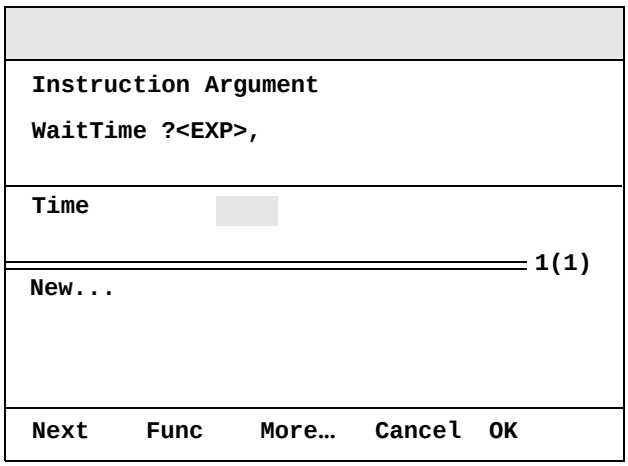


Figure 10 The dialog box used to define WaitTime.

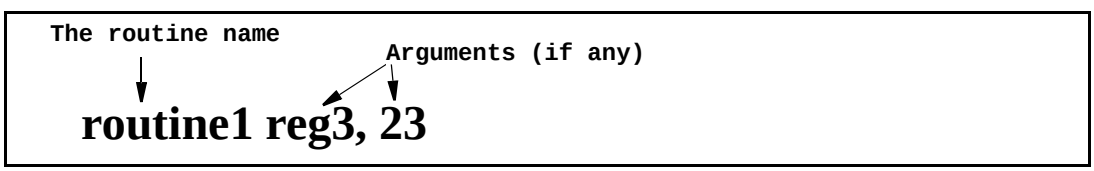
- Enter the time using the numeric keyboard.
- Press ***OK*** to confirm.

4 Controlling the Program Flow

4.1 Calling a subroutine

A call instruction contains the following information:

- information on the routine to be called,
- information on any arguments.



When this instruction is executed, the called routine will be executed. Following this, execution will return to the calling routine (see Figure 11).

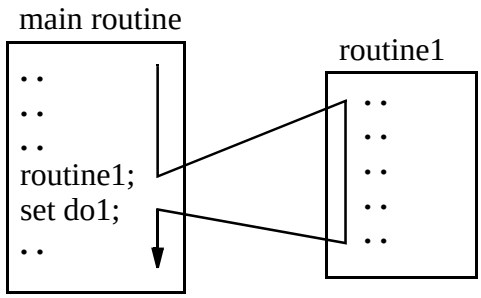


Figure 11 A routine can call another routine.

• Call up the instruction pick list for the program flow by choosing **IPL1: Prog. Flow**.

• Choose the instruction *ProcCall* by pressing the appropriate numeric key.

You must now specify the routine that is to be called. All routines are displayed for this purpose (see Figure 12).

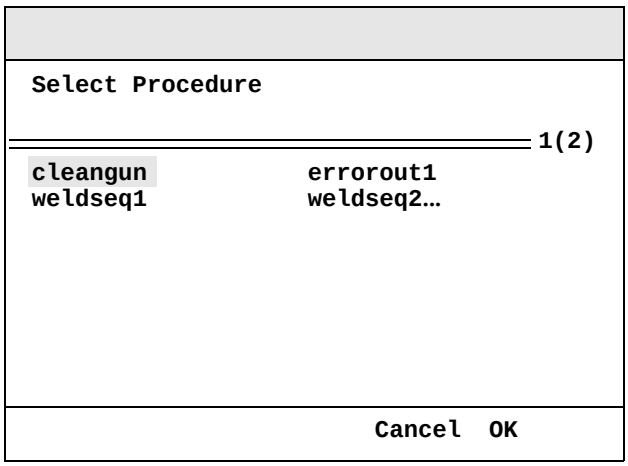


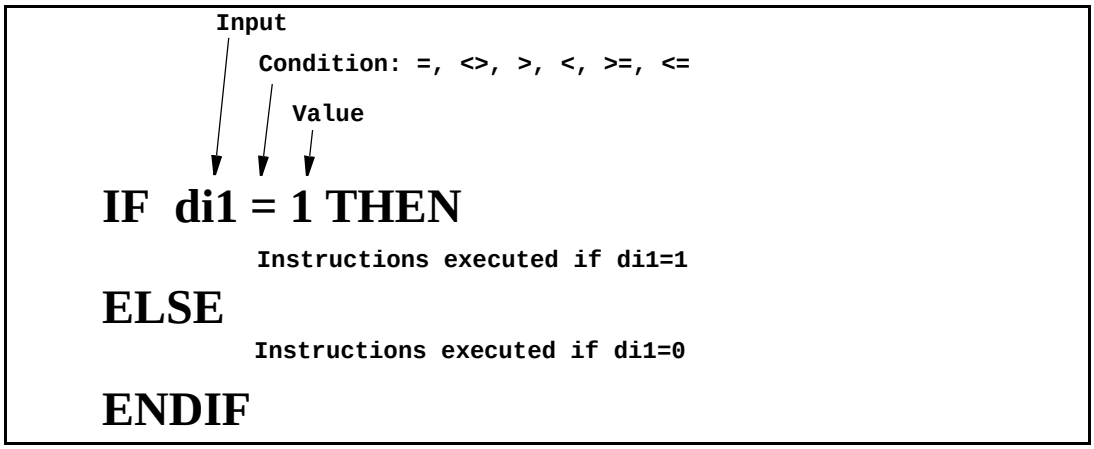
Figure 12 The dialog box used to select procedures.

• Select the desired routine and press ***OK***.

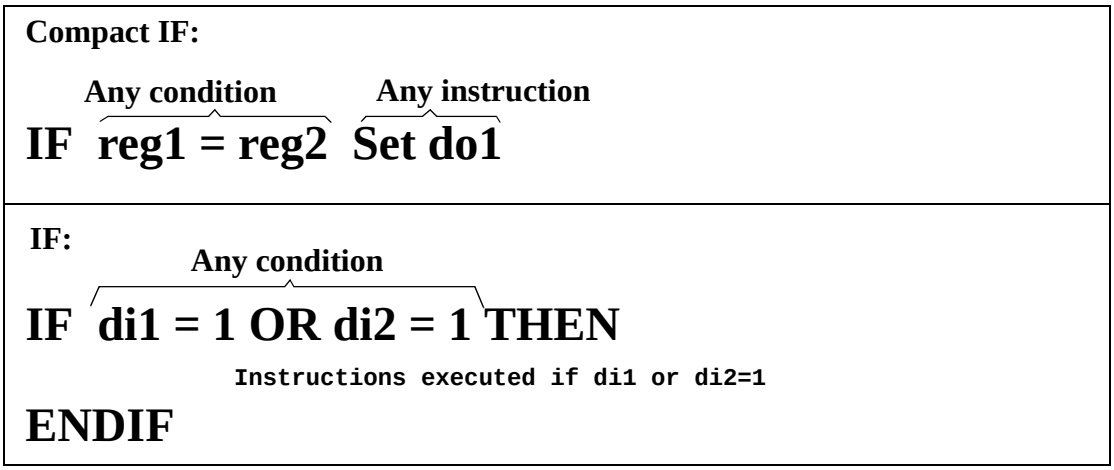
If the routine has no parameters, the instruction is ready for use; if it has parameters (indicated by ...), a dialog box will appear in which you specify the parameters of the routine in the same way as you specify an instruction argument.

4.2 Program control within a routine

The *IF* instruction is used when different instructions are to be executed depending on whether a condition is satisfied or not, e.g. depending on whether an input is set or not.



An *IF* instruction without ELSE is used when certain instructions are to be executed only if a condition is satisfied. If only one instruction is to be executed, a *Compact IF* instruction can be used.



To program an IF instruction in order to test an input

• Call up the correct instruction pick list by choosing **IPL1: Prog. Flow**.

• Choose the instruction *IF* (or *Compact IF*) by pressing the appropriate numeric key.

A dialog box will appear in which you specify the required data type for the condition (see Figure 13).

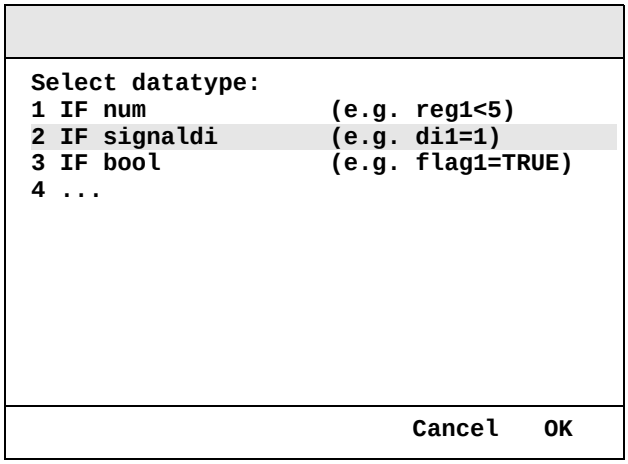


Figure 13 The dialog box used to select data type.

• Select ***IF signaldi*** and press Enter ↵. Alternatively, you can use the numeric keyboard to select the figure in front of the desired data type.

A dialog box will appear in which you can specify the desired input (see Figure 14).

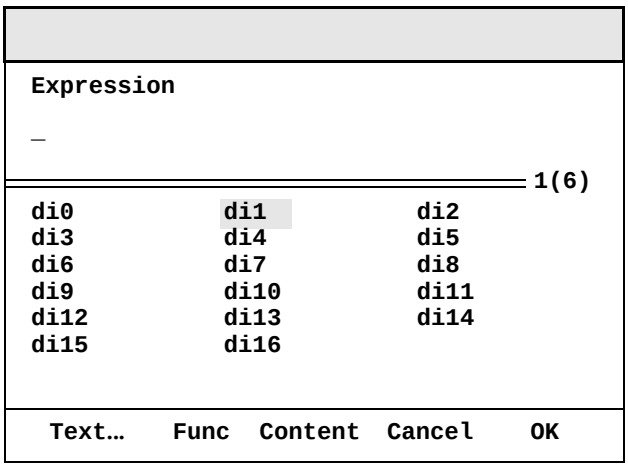


Figure 14 The dialog box used to define expression arguments.

• Select the desired input and press Enter ↵.

The dialog box used to program expressions will be called up again. All operators are now displayed in the lower part of the box.

• Select the operator = and press Enter ↵.

• Enter 0 or 1 directly using the numeric keyboard.

• Choose ***OK*** to confirm the change.

• Add instructions between THEN and ELSE and between the ELSE and ENDIF by selecting the empty instruction <SMT> and choosing the desired instructions from the pick list.

If you want to remove the ELSE part of the instruction:

• Select the complete IF instruction and press Enter ↵ .

A dialog box will appear, displaying the possible structure of the instruction. Structure parts not included in the instruction are enclosed within square brackets (see Figure 15).

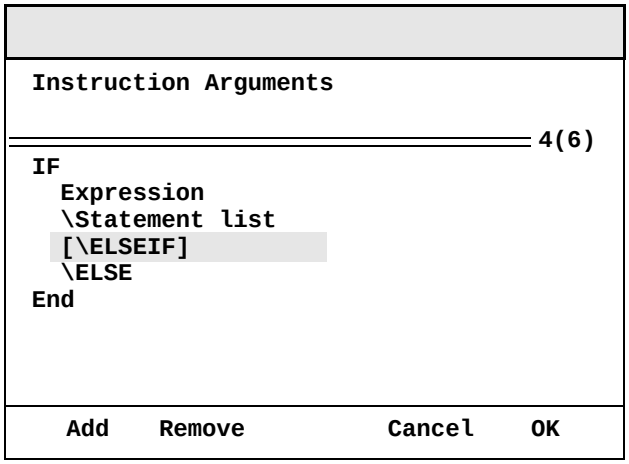


Figure 15 The dialog box to change the structure of an IF instruction.

• Select \ELSE and press ***Remove***.

• Choose ***OK*** to confirm the change.

5 Assigning a Value to Data (Registers)

An assignment instruction contains the following information:

- information on the data to be changed
- information on the desired value, which may be a complete expression, e.g. *reg1+5*reg2*.

The following instructions can be used to perform simple calculations on register variables.

Clear	reg1	clears a register
Incr	reg1	increments by 1
Decr	reg1	decrements by 1
Add	reg1, 5	adds a value (5) to a register

To program an assignment instruction

• Call up the correct instruction pick list by choosing **IPL1: Various** or **Mathematics**.

• Choose the instruction **:=** by pressing the appropriate numeric key.

You must now specify the data to be changed. All the various data are displayed for this purpose (see Figure 16).

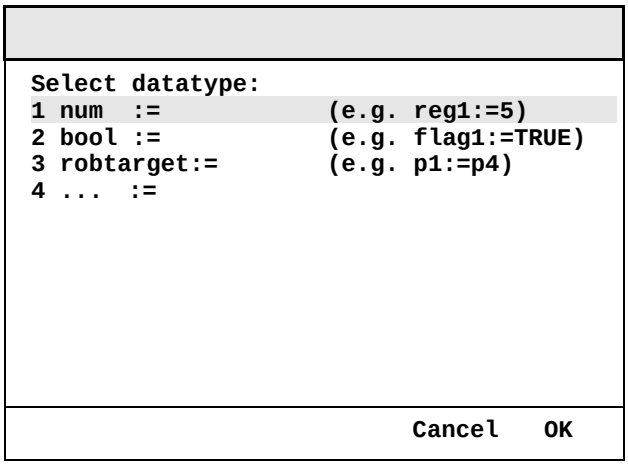


Figure 16 The dialog box used to select data type.

• Select the desired data type and press Enter ↵. Alternatively, you can use the numeric keyboard to select the figure in front of the desired data type.

If the desired data type is not found among the three predefined types, choose alternative 4 for more types. The data types that have already been used in the program will now be listed in the lower half of the box (see Figure 17).

To view all the data types, press the function key ***All***.

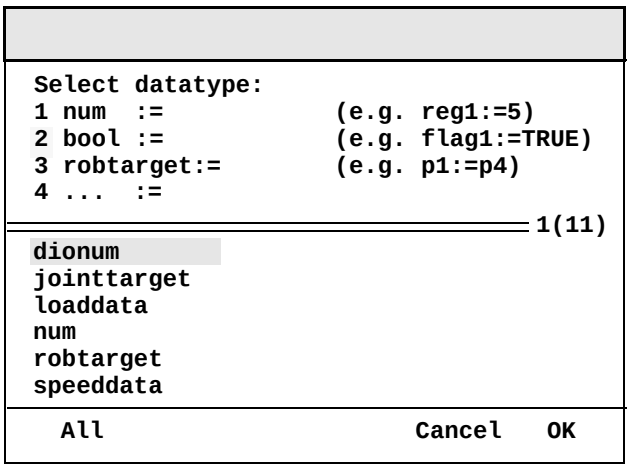


Figure 17 The dialog box shows data types used in the program.

• Choose the desired data type and press Enter ↵

A dialog box will appear in which you can define data that is to be changed (see Figure 18).

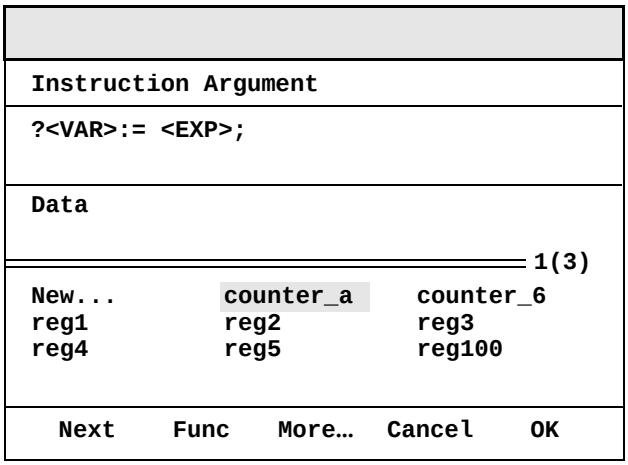


Figure 18 The dialog box used to define data that is to be changed. Only num data is shown in the list.

• Select the desired data.

• Select the next argument by pressing ***Next***.

You must now specify the new value for the data. For the purposes of this exercise, we have chosen a constant value, e.g. reg1:=5.
Use (list) to select a data instead of a numeric value.

• Using the numeric keyboard, enter the value directly.

• Choose ***OK*** to confirm the input of the instruction.

The instruction is now ready for use.

CONTENTS

Page

1 Coordinate systems 3
2 Coordinated axes 5
2.1 External axes, general 5
2.2 Coordination 5
3 Calibration 6
3.1 What is calibration? 6
3.2 Viewing the calibration status 6
3.3 Checking the calibration 7
3.4 Updating revolution counters 8
4 Base Frame for the Robot 9
4.1 Defining the Base Frame for the Robot 9
5 Coordinated track motion 12
5.1 How to get started with a coordinated track motion 12
5.2 Defining the Base Frame for a track motion 12
6 Coordinated external axes 16
6.1 How to get started with a coordinated (moveable) user coordinate system 16
6.2 Defining the User Frame for a rotational axis (single) 17
6.3 Defining the User Frame for a two-axes mechanical unit, Method 1 20
6.4 Defining the User Frame for a two-axes mechanical unit, Method 2 23
7 Defining Tools 28
7.1 Creating a new tool 28
7.2 Manually updating the TCP and weight of a tool 29
7.3 Methods of defining the tool coordinate system 29
7.4 Using the robot to change the TCP and orientation of a tool 31
7.5 Stationary tool 33
8 Work Objects and Program Displacements 35
8.1 General 35
8.2 Using work objects 36
8.3 Creating a new work object 36
8.4 Manually updating the user and object coordinate system of the work object 37
8.5 Methods of defining a work object 37
8.6 Using the robot to change the work object 38
8.7 Defining a moveable object frame 40
8.8 How to use different work objects to get different displacements 40
8.9 How to adjust the program vertically using the object frame 42
8.10 Using program displacement 42

Page

8.11 Creating a new displacement frame..... 43
8.12 Manually updating a displacement frame 43
8.13 Methods for defining a displacement frame..... 44
8.14 Using the robot to change a displacement frame 44

Calibration

1 Coordinate systems

All robot positions in a robot program, are stored in rectangular coordinates (e.g. xyz values for position), related to a defined coordinate system (or frame). This coordinate system may in turn be related to another coordinate system etc. in a chain. Some of these coordinate systems are embedded in the configuration of the robot system, and are not visible to the user, while others may be programmed by the user. The table below provides an overview of the various coordinate systems (or frames) used in the robot system:

Coordinate system	Defined where	Related to
Base Frame of robot	Service/View:BaseFrame. Base frame definition of robot gives relation between world and base frame.	World Frame
World Frame	No definition needed	Nothing
User Frame, fixed in room. (Tool mounted on robot)	Program/View: Data Types - wobjdata In any work object data	World Frame
User Frame, fixed on robot mounting plate. (Tool fixed in room)	Program/View: Data Types - wobjdata In any work object data	Wrist Frame
User Frame, coordinated to an external axis	Service/View:/BaseFrame. In the base frame definition of an external mechanical unit	World Frame
Object Frame	Program/View: Data Types - wobjdata In any work object data	User Frame
Program Displacement Frame	In the system variable C_PROGDISP, set up by instructions PDispSet or PDispOn etc.	Object Frame
Robtarget frame (Programmed position)	When a position is programmed.	Program Displacement Frame
Base Frame of a mechanical unit (only for internal system use)	Service/View:BaseFrame. In the base frame definition of an external mechanical unit or as configuration parameter.	World Frame
Wrist Frame	Implicit in the kinematic model of robot	Base Frame of the robot.
Tool Frame (Tool mounted on robot)	Program/View: Data Types - tooldata In any tool data	Wrist Frame
Tool Frame (Tool fixed in room)	Program/View: Data Types - tooldata In any tool data	World Frame

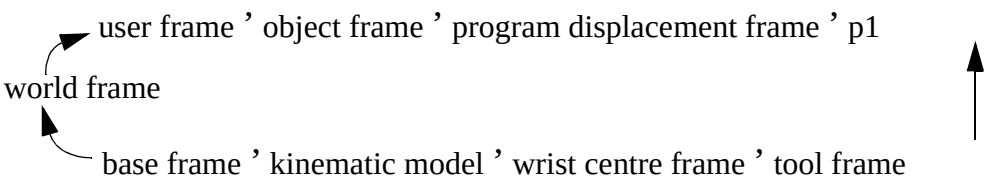
Now any programmed position, e.g. p1, will be related to the World Coordinate system through the chain:

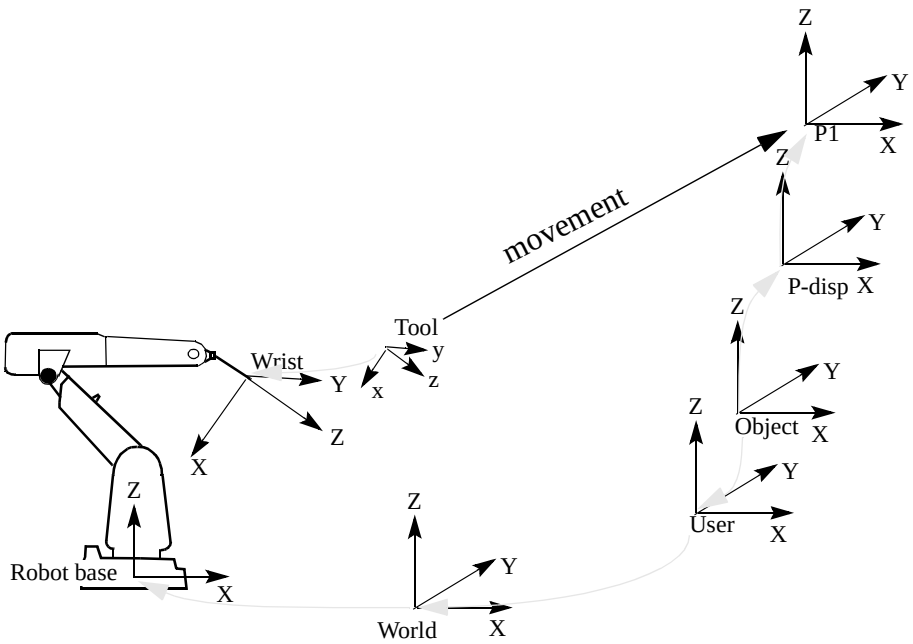
world frame ’ user frame ’ object frame ’ program displacement frame ’ p1

The current position of the robot, i.e. the location of the tool, is related to the World Coordinate system through the chain:

world frame ’ base frame ’ kinematic model ’ wrist centre frame ’ tool frame

When the robot is moved in automatic mode to a programmed position, the aim is to bring the tool (tool frame) to coincide with the programmed position, i.e. to close the chain:


user frame ’ object frame ’ program displacement frame ’ p1
world frame
base frame ’ kinematic model ’ wrist centre frame ’ tool frame



The accuracy of the robot, i.e. how well the tool frame will coincide with the programmed position, is normally independent of the accuracy of the various coordinate systems. This is true, however, only if the same coordinate systems are used as when programming the robot, pointing out all positions with the robot (repetition accuracy). If the coordinate systems are changed, making it possible to displace the program, then the accuracy is dependent on every single link in the chain. This means that the accuracy is directly dependent on the calibration accuracy of the various frames. This is even more important for off-line programming.

In the following chapters, an overview will be given of the steps to be taken to calibrate and define the robot and the different coordinate systems mentioned above.

2 Coordinated axes

2.1 External axes, general

All external axes are handled in mechanical units. This means that before an external axis may be moved, the mechanical unit to which it belongs, must be activated. Within a mechanical unit, the different axes will be given a logical name, from *a* to *f*. In the system parameters, these logical axes will be connected to the external axes joints. For each joint a motor and a drive unit is defined. Different joints may share the same motor and drive unit.

Two or more mechanical units may be activated at the same time, as long as they do not have the same logical axes defined in their set of external axes. However, two or more mechanical units may have the same logical axes, if they are not activated simultaneously.

Two or more mechanical units may not be activated at the same time, if they share one or more drive units, even if they use separate logical axes. This means that two logical axes, each belonging to different mechanical units, may control the same drive unit, but not at the same time.

2.2 Coordination

A mechanical unit may be coordinated or not coordinated with the robot movements.

If it is not coordinated, each axis will be moved independent of the robot movements, e.g. when jogging, only the separate axis will move. However during program execution, the external axes will be synchronized to the robot movement, in such a way that both movements will be completed in the same time.

If the mechanical unit is coordinated, it is guaranteed that the robot TCP movements, as seen in the object or user coordinate system, will be the same irrespective of the movements of the external axes.

Two types of coordination categories exist. The first category of coordination is when the robot itself is moved, e.g. the coordination to a gantry or track movement. This means that the robot is mounted on a gantry or a track, and may be moved along these axes. The world and user/object coordinate systems, however, will be fixed in the room, and the robot movements in these coordinate systems will be independent of simultaneous gantry or track movements. This coordination is automatically active, if the mechanical unit with the track motion is active.

The second coordination category, is when the robot movements are coordinated to the movements of a user frame connected to a mechanical unit. E.g. a user frame may be placed on a turntable and connected to its movements. An ordinary work object may be used for this purpose, if it is marked with the name of the mechanical unit to be connected to, and that it should be moveable. The coordination will be active if the mechanical unit is active, and the “coordinated” work object is active. When such a “coordinated” work object is used, in jogging or in a move instruction, the data in the “uframe” component will be ignored and the location of the user coordinate system will

only depend on the movements of the mechanical unit. However the “oframe” component will still work giving an object frame related to the user frame and also the displacement frame may be used.

3 Calibration

3.1 What is calibration?

Calibration involves setting the calibration positions (zero positions) of the axes and is used as the basis for their positioning. If the robot or external axes are not correctly calibrated, this will result in incorrect positioning and will have a negative effect on the agility of the robot. The robot is calibrated on delivery.

The position of the robot axes is determined using a resolver and a counter that counts the number of resolver revolutions. If the robot is correctly calibrated, it is automatically able to calculate the current position on start-up.

Calibration is carried out in two stages:

- Calibration of resolvers (fine calibration): the axes are placed in their specific calibration positions and the current resolver values are stored. For information on how to do this, see the chapter on *Repairs* in the Product Manual.
- Update of revolution counters: the correct motor revolution for the calibration is defined; the axes are placed close to their calibration positions and the revolution counters are updated.

The position of an external axis is determined using sync. switches. The same method used for the robot can be used.

3.2 Viewing the calibration status

• Press the Miscellaneous key and select the Service window.

• Choose **View: Calibration.**

This window displays an overview of the status of all the mechanical units in the robot system (see Figure 1).

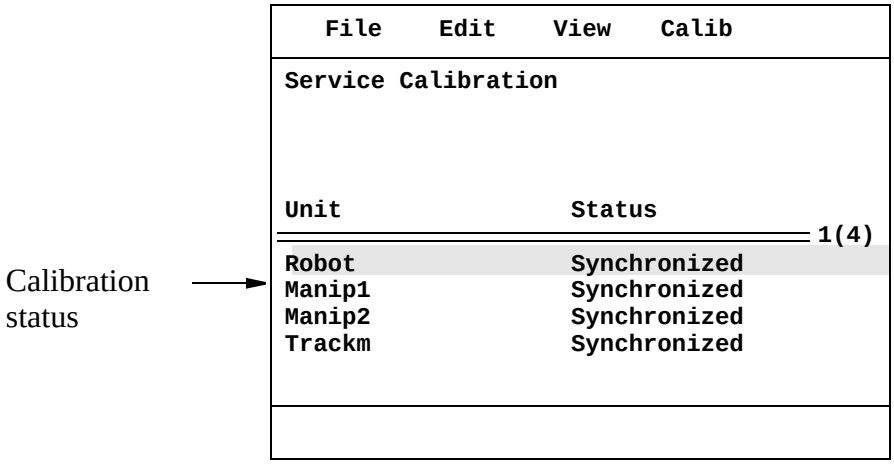


Figure 1 The Service Calibrate window shows whether or not the robot system is calibrated.

The calibration status can be any of the following:

- **Synchronized**
  All axes are calibrated and their positions are known. The unit is ready for use.
- **Not updated Rev. Counter**
  All axes are fine-calibrated but one (or more) of the axes has a revolution counter that is NOT updated. This or these must thus be updated.
- **Not calibrated**
  One (or more) of the axes is NOT fine-calibrated. This or these must thus be fine-calibrated.
- **Unsynchronized**
  At least one of the axes has a position that is NOT known. An external axis with a sync. switch must thus be synchronized. See Section 5, *Starting up*, in this manual.

3.3 Checking the calibration

If a revolution counter is incorrectly updated, it will cause incorrect positioning. Thus, check the calibration very carefully after each update.
An incorrect update can damage the robot system or injure someone.

- Run the calibration program under the /SERVICE/CALIBRAT/ directory on the system diskette, *Set up*. An alternative method is to jog the robot axis-by-axis until the axis angles in the Jogging window equal zero.
- Check each axis to see if the marks are positioned exactly opposite one another. If they are not, the calibration must be redone.

The marks may be scribed lines, vernier scales or the like. Their location is described in the chapter on *Installation and Commissioning* in the Product Manual.

3.4 Updating revolution counters

- Open the Service window.
- Choose **View: Calibration**.
- Select the desired unit.
- Move the robot or the chosen unit close to (half a motor revolution at the furthest) the calibration pose. The latter is usually indicated by a scribed line or a vernier scale. The calibration pose of the robot is described in the chapter on *Installation and Commissioning* in the Product Manual.
- Choose **Calib: Rev.Counter Update.**

A dialog box will appear, in which you can choose the axis you want to update (see Figure 2).

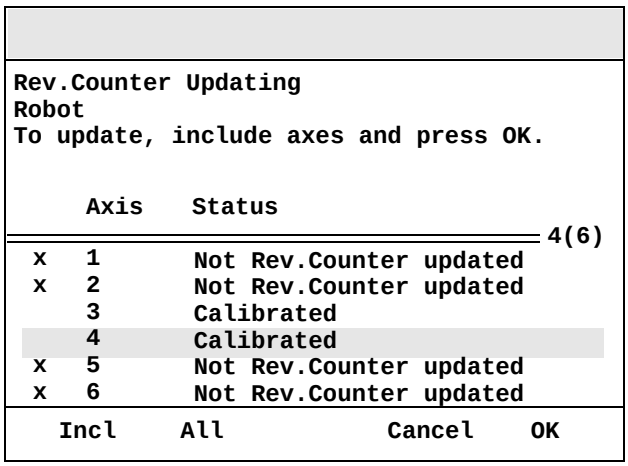


Figure 2 The dialog box used to select axes when updating the revolution counter.

- Select the axis to be updated and press the ***Incl*** function key.
  An **x** to the left indicates that the axis is to be updated.
- Use the same procedure on the remaining axes or press the function key ***All*** which selects all axes. A selected axis can be deselected by pressing the ***Excl*** function key.
- Confirm the choice of axes by pressing ***OK***.
- Start updating by pressing ***OK*** in the confirmation dialog box.

4 Base Frame for the Robot

4.1 Defining the Base Frame for the Robot

The following methods are used to define the location of the robot's base frame in relation to the world coordinate system.

In order to define a robot base frame you need a world fixed tip within the robot's working range, and optionally an elongator attached to the tip. If the robot is mounted on a track or similar, the track should be in its calibration position. The calibration procedure consists of a number of positionings for the robot's TCP to a reference point. The reference point´s coordinates in the world coordinate system, must be known. The coordinates must be stated before the calibration can be done.

The following positions on the world fixed tip device are involved in the calibration:

- the tip itself (with known coordinates in world), used when defining the base frame translation
- one point on the elongator defining the positive z direction for the world coordinate system
- one point on the elongator defining the positive x direction for the world coordinate system.

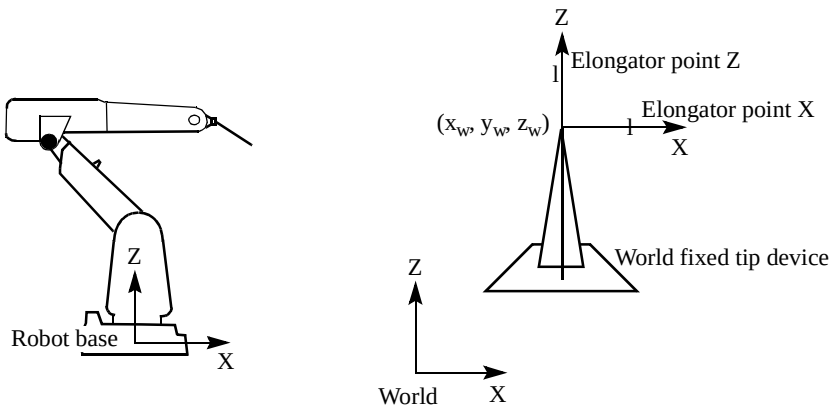


Figure 3 Robot base frame definition points.

When the necessary conditions are fulfilled the definition of the robot base frame can be performed. Please observe, that in the case of a track mounted robot, the track must be in the calibration position before the base frame of the robot may be defined.

• Press the Miscellaneous key and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the robot and press Enter or ***Def***.

A dialog like the one in Figure 4 will appear.

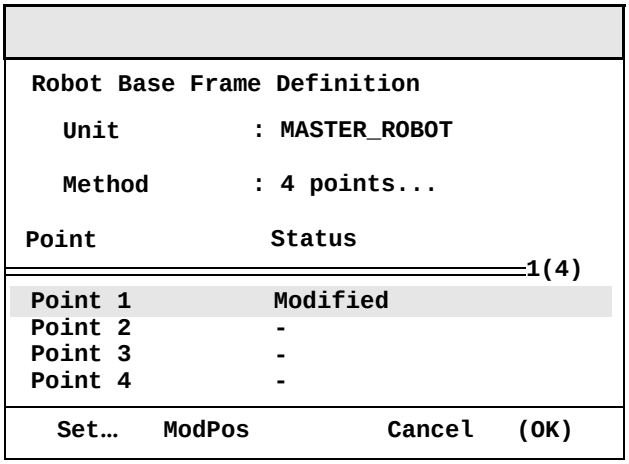


Figure 4 Robot base frame definition dialog.

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed.

• Select the field ***Method*** and press Enter ↵.
• Choose method for definition and press ***OK***.

The method requires an elongator attached to the world fixed tip.

Input of world coordinates of the reference point

• Press ***Set***.
• Input the x, y and z values.
• Verify that the input is correct and press ***OK***.

To record world fixed reference points

• Select the first point ***Point 1***.
• Jog the robot as close as possible to the world fixed tip.
• Modify the position by pressing the function key ***ModPos***.
• Repeat the above for the points ***Point 2*** to ***Point n***.

To record the elongator X point

• Select the elongator point ***Point X***.
• Jog the robot as close as possible to the elongator point on the positive X axis.
• Modify the position by pressing the function key ***ModPos***.

To record the elongator Z point

• Select the elongator point ***Point Z***.

• Jog the robot as close as possible to the elongator point on the positive Z axis.

• Modify the position by pressing the function key ***ModPos***.

To calculate the robot base frame

• Press ***OK*** to calculate the robot base frame for the selected mechanical unit.

When the calculation is finished, a dialog like the one in Figure 5 will appear.

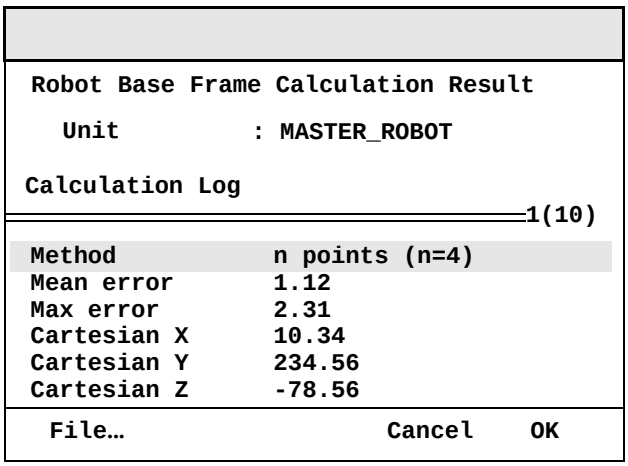


Figure 5 The result of a robot base frame calculation.

Field	Description
Unit	The name of the mechanical unit for which the definition of robot base frame is to be done.

List contents	Description
Method	Displays the selected calibration method.
Mean error	The accuracy of the robot positioning against the tip.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the base frame.
Cartesian Y	The y coordinate for the base frame.
Cartesian Z	The z coordinate for the base frame.
Quaternion 1-4	Orientation components for the base frame.

The result of the calculation is expressed in the world coordinate system.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new robot base frame.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the Service window to activate the base frame.

The definition is now complete, but before proceeding with other tasks, verify it by jogging the robot in the world coordinate system.

5 Coordinated track motion

5.1 How to get started with a coordinated track motion

In the checklist below, the steps required to coordinate track motion are described. In each step, there may be a reference to another chapter in this manual, where more details of the specific actions to be taken will be found.

• Define the system parameters for the track motion, see chapter 12 in this manual System Parameters/*Defining a track motion with coordinated motion*. Find out the name of this mechanical unit, and the corresponding logical axis.

• Calibrate the robot and the track motion, i.e. the zero position of the measuring system for both robot and track must be carefully determined. See *Calibration* on page 6.

• Define the base frame of the robot, see *Defining the Base Frame for the Robot* on page 9. Please observe that the track must be in its calibration position when the robot base frame is defined.

• Define the base frame of the track, see *Defining the Base Frame for a track motion* on page 12.

• Store all these definitions on a diskette by giving the command **File: Save All as** in the System parameter window. See chapter 12 in this manual.

• Activate the track unit in the jogging window and check that the coordination is working satisfactorily. This may be done by choosing ***World*** or ***Wobj*** in the field ***Coord*** and then jogging the track axis. The robot TCP should not move, but be fixed relative to the object coordinate system.

5.2 Defining the Base Frame for a track motion

To make coordinated track motion possible it is necessary to define the base frame of the track. This frame is located in the calibration position of the track, see Figure 6.

For the definition of a track base frame you need a world fixed tip within the robot's working range. The calibration procedure consists of a number of positionings of the TCP to the reference point. Please note that before the base frame of the track may be defined, the base frame of the robot must be defined with the track in the calibration position.

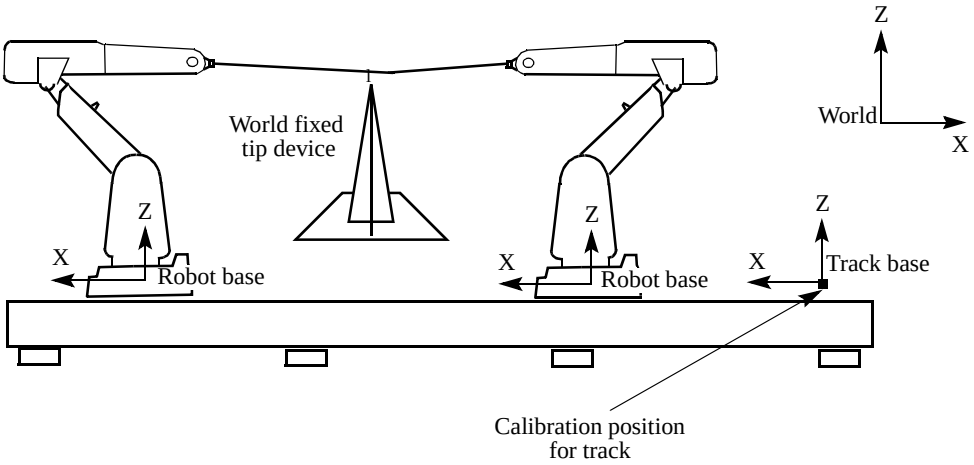


Figure 6 Track base frame definition procedure.

The track's base coordinate system has its origin in the robot's base when the track is in its calibration position. The x direction is pointing along the linear track path and the z axis of the track's coordinate system is parallel with the z axis of the robot's base coordinate system.

Figure 7 shows an example of how the base systems are oriented for a specific robot mounting. In this case the robot is mounted on the track at an angle of 45 degrees.

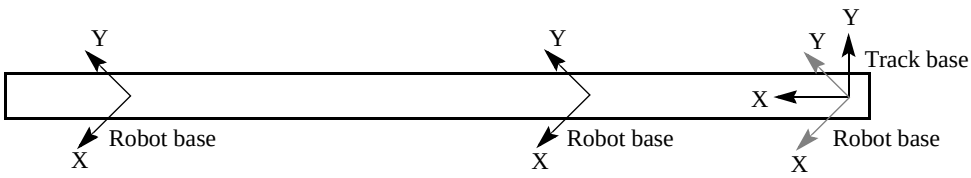


Figure 7 Track and robot base coordinate systems seen from above.

• Press the Miscellaneous key and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the track unit and press Enter or ***Def***.

A dialog like the one in Figure 8 will appear.

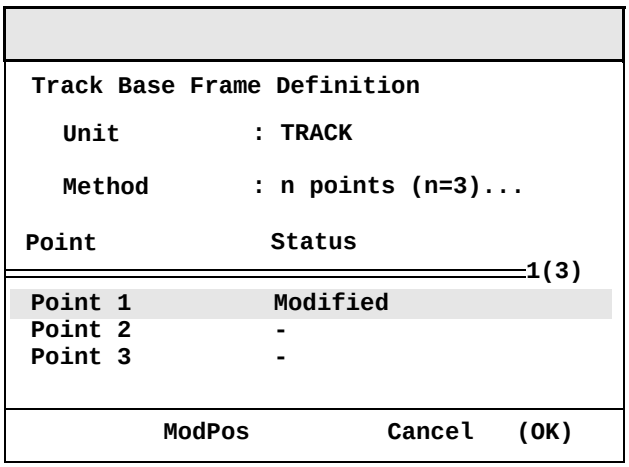


Figure 8 Track base frame definition dialog.

To choose definition methods

Before you start modifying any positions, make sure the desired method is displayed. The method defines the number of track positions from where the robot TCP will be moved to the reference point.

- Select the field ***Method*** and press Enter ↵.
- Choose the number of points to be used for definition and press ***OK***. (Currently only the three point method is implemented.)

To record world fixed reference points

Activate the track unit and run it to the calibration position, i.e. zero position should be displayed on the teach pendant.

- Select the first point ***Point 1***.
- Jog the robot as close as possible to the world fixed tip.
- Modify the position by pressing the function key ***ModPos***.
- Move the robot along the track and repeat the steps above for the points ***Point 2*** and ***Point 3***.

To calculate the track base frame

- Press ***OK*** to calculate the track base frame for the selected mechanical unit.

When the calculation is finished, a dialog like the one in Figure 9 will appear.

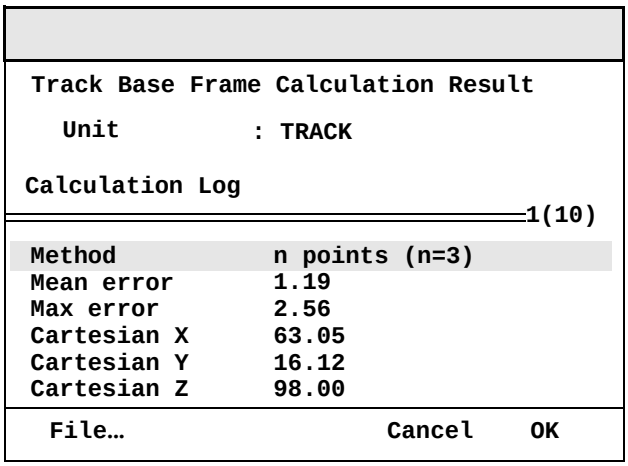


Figure 9 The result of a track base frame calculation.

The result of the calculation is expressed in the world coordinate system.

Field	Description
Unit	The name of the mechanical unit for which the definition of base frame is to be done.

List contents	Description
Method	Displays the selected track definition method.
Mean error	The accuracy of the robot positioning against the tip.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the base frame. (x, y, z is the same as for the robot base frame).
Cartesian Y	The y coordinate for the base frame.
Cartesian Z	The z coordinate for the base frame.
Quaternion 1-4	Orientation components for the base frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new track base frame.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the Service window to activate the track base frame.

The definition is now complete but before proceeding with other tasks, verify it by doing the following:

- Point out with the robot, in coordinated mode, the world fixed reference point with the track in different positions, and print out the position in world coordinates. Jog the track in coordinated mode.

6 Coordinated external axes

6.1 How to get started with a coordinated (moveable) user coordinate system

In the checklist below, the steps required to coordinate a user coordinate system are described. In each step, there may be a reference to another chapter in this manual, where more details of the specific actions to be taken will be found.

- Define the system parameters for the external mechanical unit, see chapter 12 in this manual System Parameters/*Defining an external mechanical robot coordinated with the robot*. Find out the name of this mechanical unit, and the corresponding logical axis.
- Calibrate the robot and the mechanical unit, i.e. the zero position of the measuring system for both robot and mechanical unit must be carefully determined. See *Calibration* on page 6.
- Define the base frame of the robot, see *Defining the Base Frame for the Robot* on page 9.
- Define the user frame of the mechanical unit, see *Defining the User Frame for a rotational axis (single)* on page 17 or *Defining the User Frame for a two-axes mechanical unit, Method 1* on page 20 or *Defining the User Frame for a two-axes mechanical unit, Method 2* on page 23.
- Store all these definitions on a diskette, by giving the command **File: Save All as** in the System parameter window. See chapter 12 in this manual.
- Create a new work object data and give it a name, e.g. *turntable*. In this work object, change the component *ufprog* to FALSE, indicating that the user object should be connected to a moveable mechanical unit. Also change the component *ufmec* to the name of the mechanical unit *turntable* (must be written in text mode).
- If you want the object frame to be displaced relative to the user frame, you may write the displacement in the x, y, z values of the “oframe” component of the work object. For other methods see *Defining a moveable object frame* on page 40.
- Activate the mechanical unit in the jogging window and check that the coordination is working satisfactorily. This may be done by choosing ***Wobj*** in the field ***Coord***, and the work object, e.g. *turntable*, in the field ***Wobj***, and then jogging one of the mechanical unit axes. The robot TCP should also move, following the moveable object coordinate system.
- When programming, it is important to have the coordinated work object, in this case *turntable*, programmed as an argument in each move instruction. This will be automatically added to the move instruction, if the work object is activated in the jogging window before starting the programming.

6.2 Defining the User Frame for a rotational axis (single)

This method will define the location of the user coordinate system of a rotational single axis type mechanical unit, relative to the world coordinate system. This user coordinate system should be used when a coordinated work object is used.

The definition of a user frame for a rotational external axis requires that the turntable on the external axis has a marked reference point. The calibration procedure consists of a number of positionings for the robot's TCP on the reference point when the turntable is rotated to different angles. See Figure 10.

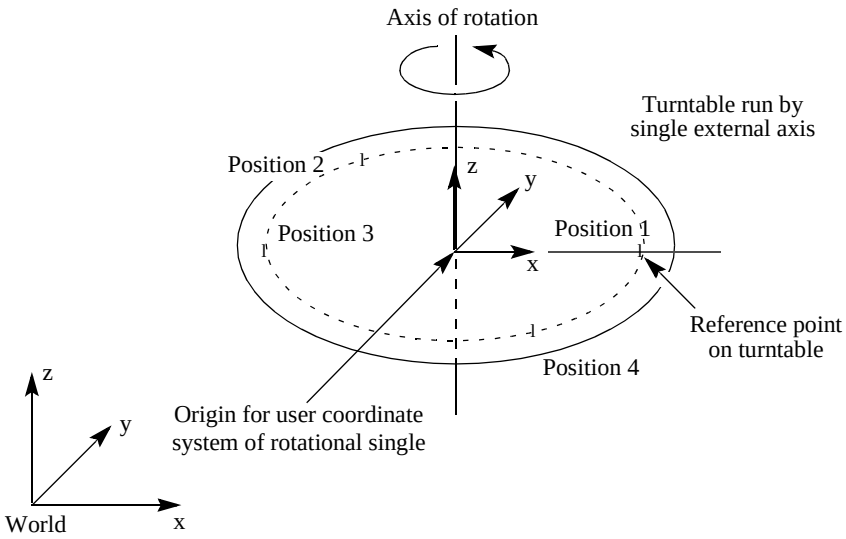


Figure 10 Definition points for a rotational axis.

The user coordinate system for the rotational axis has its origin in the centre of the turntable. The z direction coincides with the axis of rotation and the x axis goes through the reference point. Figure 11 shows the user coordinate system for two different positionings of the turntable (turntable seen from above).

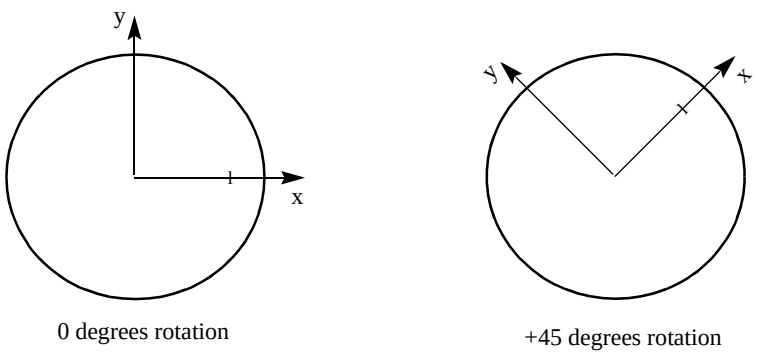


Figure 11 The user coordinate system at various angles of rotation.

• Press the Miscellaneous key and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the mechanical unit and press Enter ↵ or ***Def***.

A dialog like the one in Figure 12 will appear.

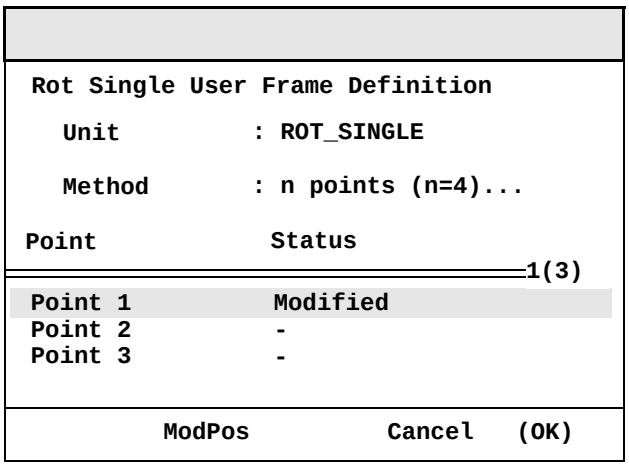


Figure 12 Dialog for definition of user frame for a rotational axis.

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed.

• Select the field ***Method*** and press Enter ↵.

• Choose number of points to use for definition and press ***OK***. (Currently only the four point method is implemented.)

To record turntable reference points

Activate the mechanical unit and run it to its calibration position, i.e. zero position should be displayed on the teach pendant.

• Select the first point ***Point 1***.

• Point out the reference point on the turntable with the robot's TCP.

• Modify the position by pressing the function key ***ModPos***.

• Rotate the turntable in the positive direction and repeat the above for the points ***Point 2*** and ***Point 3***.

To calculate the user frame

• Press ***OK*** to calculate the user frame for the selected mechanical unit.

When the calculation is finished a dialog like the one in Figure 13 will appear.

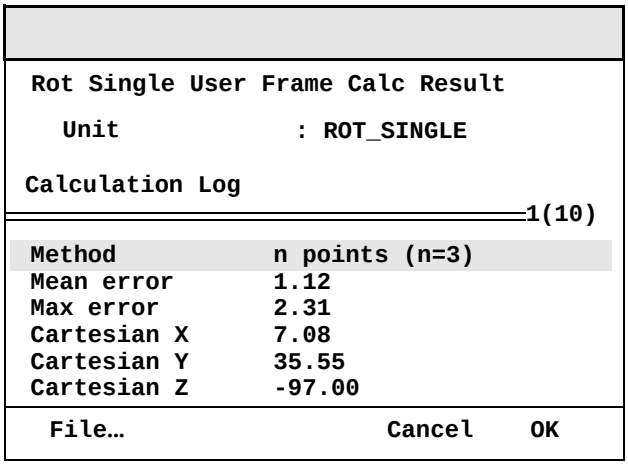


Figure 13 The result of a user frame calculation for a rotating single.

The calculation log shows the user frame expressed in the world coordinate system when the mechanical unit is in its calibration position.

Field	Description
Unit	The name of the mechanical unit for which the definition of user frame is to be done.
List contents	Description
Method	Displays the selected calibration method.
Mean error	The accuracy of the robot positioning against the reference point.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the user frame.
Cartesian Y	The y coordinate for the user frame.
Cartesian Z	The z coordinate for the user frame.
Quaternion 1-4	Orientation components for the user frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new user frame.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the Service window to activate the user frame.

The definition is now complete, but before proceeding with other tasks, verify it by jogging the mechanical unit in coordinated mode.

Note The user frame is stored in the system parameters as the base frame of the external mechanical unit. The user frame in the corresponding work object is therefore not used.

6.3 Defining the User Frame for a two-axes mechanical unit, Method 1

This method will define the location of the user coordinate system of an “Orbit” type mechanical unit, relative to the world coordinate system. This user coordinate system should be used when a coordinated work object is used.

It should be noted that this method requires that the kinematics (relationship between two axes) of the mechanical unit are defined in the robot system configuration. Therefore, this method can only be used for workpiece manipulators supplied by ABB, where a ready-made configuration was included in the delivery. For other types of workpiece manipulator see *Defining the User Frame for a two-axes mechanical unit, Method 2* on page 23.

The definition of this user coordinate system requires that the orbit turntable is marked with a coordinate system as shown in Figure 14. The coordinate system must have the x axis in the plane of the two turning axes of the Orbit station, when the turn table is in its calibration position.

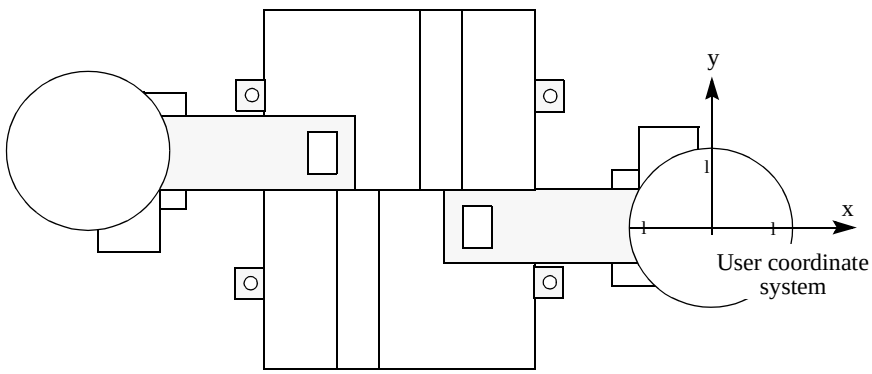


Figure 14 Orbit user coordinate system.

The coordinate system of the orbit station has its xy plane in the surface of the turntable, and the origin is located in the centre of the turntable, i.e. the z axis will coincide with the second axis.

• Press the Miscellaneous key and select the Service window.

• Choose **View:BaseFrame**.

A dialog containing all synchronized mechanical units is shown.

• Select the mechanical unit and press Enter ↵ or ***Def***.

A dialog like the one in Figure 15 will appear.

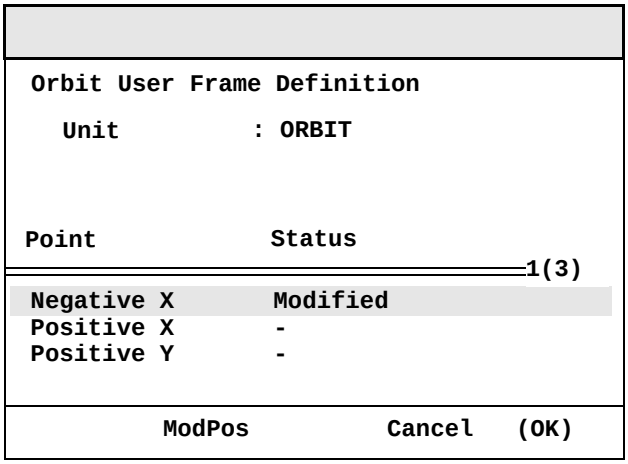


Figure 15 Dialog for definition of user frame for orbit station.

To record reference points

• Activate the mechanical unit and run it to its calibration position, i.e. zero position should be displayed on the teach pendant.

• Select the first point ***Negative X***.

• Point out the reference point on the negative x axis with the robot's TCP (it is not necessary that the position is on the negative side of the origin, but it must be on the negative side relative to the next point "Positive X").

• Modify the position by pressing the function key ***ModPos***.

• Select the point ***Positive X***.

• Point out the reference point on the positive x axis with the robot's TCP.

• Modify the position by pressing the function key ***ModPos***.

• Select the point ***Positive Y***.

• Point out the reference point on the positive y axis with the robot's TCP.

• Modify the position by pressing the function key ***ModPos***.

To calculate the user frame

• Press ***OK*** to calculate the user frame for the selected mechanical unit.

When the calculation is finished, a dialog like the one in Figure 16 will appear.

The calculation log shows the user frame expressed in the world coordinate system when the mechanical unit is in its calibration position.

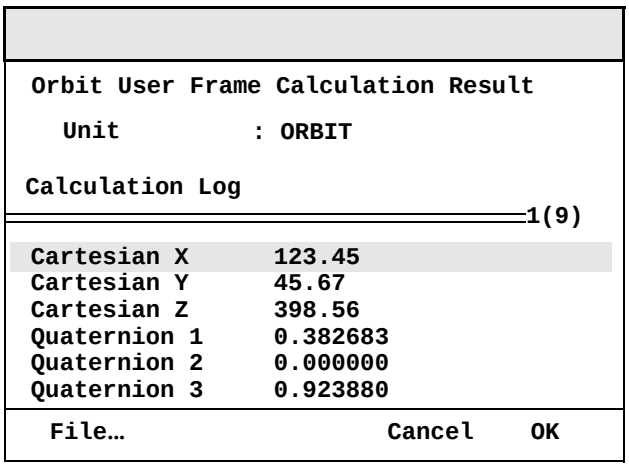


Figure 16 The result of a linear moving base frame calculation.

Field	Description
Unit	The name of the mechanical unit for which the definition of user frame is to be done.
List contents	Description
Cartesian X	The x coordinate for the user frame.
Cartesian Y	The y coordinate for the user frame.
Cartesian Z	The z coordinate for the user frame.
Quaternion 1-4	Orientation components for the user frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new user frame.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the *Service* window to activate the user frame.

The definition is now complete, but before proceeding with other tasks, verify it by jogging the mechanical unit in coordinated mode.

Note The user frame is stored in the system parameters as the base frame of the external mechanical unit. The user frame in the corresponding work object is therefore not used.

6.4 Defining the User Frame for a two-axes mechanical unit, Method 2

This method will define the location of the user coordinate system of an “Orbit” type mechanical unit, relative to the world coordinate system. This user coordinate system should be used when a coordinated work object is used.

It should be noted that this method does not require that the kinematics (relationship between two axes) of the mechanical unit are defined in the robot system configuration. If this is a known factor, another method can be used. See *Defining the User Frame for a two-axes mechanical unit, Method 1* on page 20.

Figure 17 shows an orbit station with two rotational axes and a turntable mounted on the second axis.

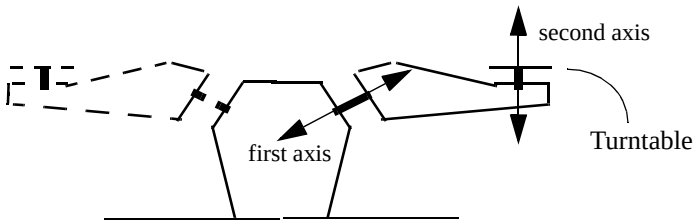


Figure 17 Geometric structure of an orbit station.

The definition of the user frame requires that the turntable has a marked reference point. The origin of the user frame is located in the centre of the turntable with the z axis coinciding with the second axis of rotation. The x axis goes through the reference point (see Figure 18).

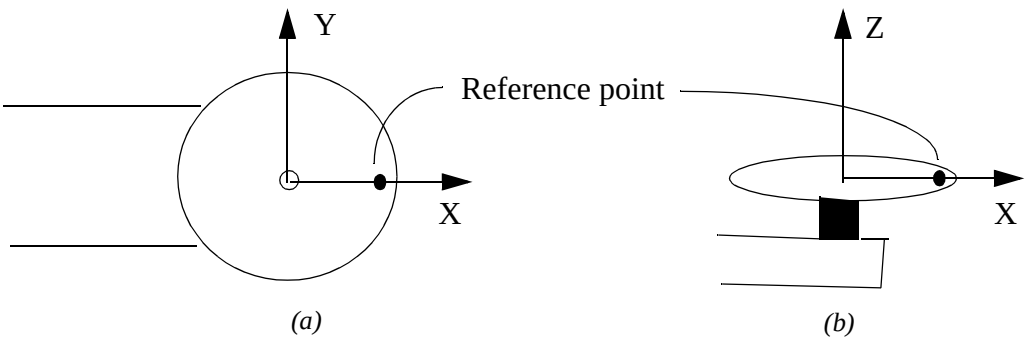


Figure 18 The turntable seen from above (a) and side (b).

The user frame is determined by two definition procedures. One procedure for the first axis and another similar procedure for the second axis. These two procedures are performed separately but both are necessary to complete the user frame definition.

• Press the Miscellaneous key and select the Service window.

• Choose **View: Two Axes Definition**

A dialog containing all synchronized mechanical units is shown.

• Select the mechanical unit and press Enter or ***Def***.

A dialog like the one in Figure 19 will appear.

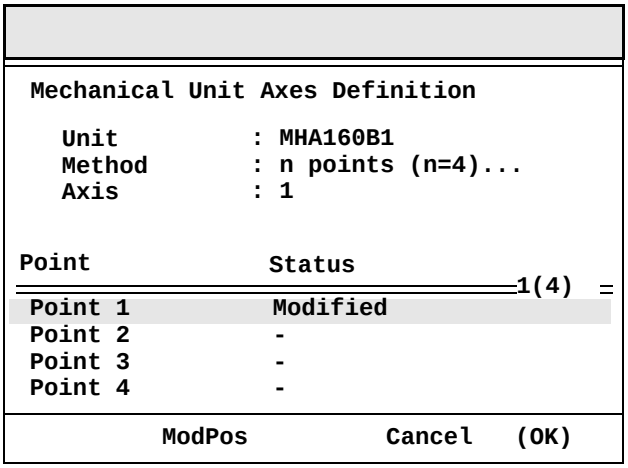


Figure 19 Dialog for definition of axes.

Defining the first axis

Before defining the first axis, both axes must be run to their calibration positions. The procedure to define the first axis consists of a number of positionings for the robot's TCP on the reference point when the first axis is rotated to different angles. Position 1 is the position of the reference point when both axes are fixed to their calibration positions. The following positions, position 2, 3, 4 etc., are the positions of the reference point when the first axis is rotated to greater angles in successive steps. See Figure 20.

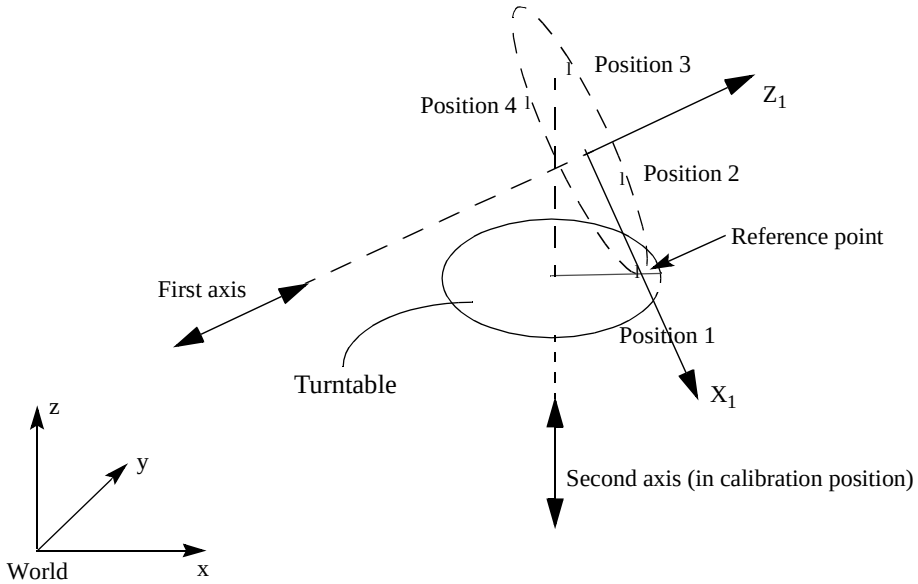


Figure 20 Definition of the first axis. Four positionings of the robot's TCP on the reference point are performed with the first axis rotated to different angles.

Defining the second axis

Before defining the second axis, both axes must be run to their calibration positions. The procedure to define the second axis consists of a number of positionings for the robot's TCP on the reference point when the second axis is rotated to different angles. Position 1 is the position of the reference point when both axes are fixed to their calibration positions. The following positions, position 2, 3, 4 etc., are the positions of the reference point when the second axis is rotated to greater angles in successive steps. See Figure 21.

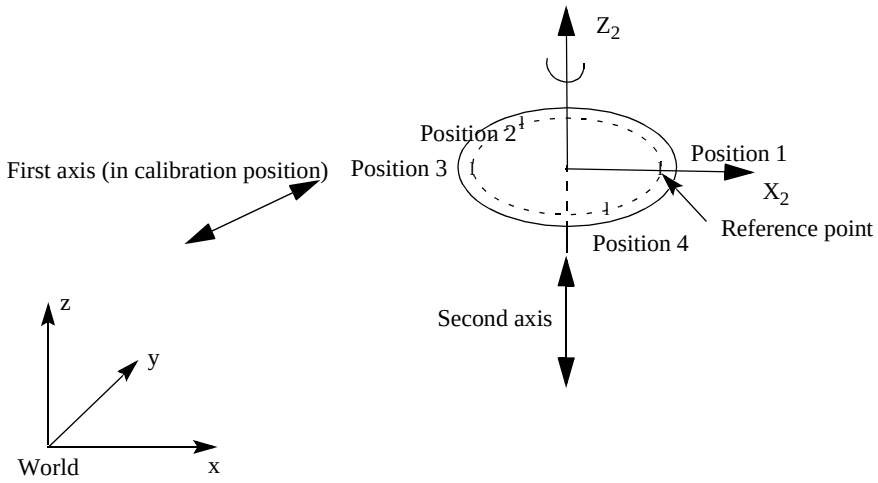


Figure 21 Definition of the second axis. Four positionings of the robot's TCP on the reference point are performed with the second axis rotated to different angles.

This frame coincides with the user frame when both axes are fixed to their calibration positions.

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed and the mechanical unit is activated.

• Select the field ***Method*** and press Enter ↵.

• Choose the number of points to use for the axis definition and press **OK**.

To choose axis

You can choose which one of the axes you want to define. Remember that both axes must be defined to complete the user frame definition. It is possible to redefine both axes or just one of them.

• Select the field ***Axis*** and press Enter ↵ to switch axis.

To record reference points for the first axis definition

Make sure Axis 1 is chosen. Run the mechanical unit to its calibration position.

- Select the first point, ***Point 1***.
- Point out the reference point on the turntable with the robot's TCP.
- Modify the position by pressing the function key ***ModPos***.
- Rotate the first axis to a greater angle and repeat the above for the points ***Point 2*** to ***Point n***.
- Press ***OK*** to calculate the frame of the first axis.

To record reference points for the second axis definition

Make sure Axis 2 is chosen. Run the mechanical unit to its calibration position.

- Select the first point ***Point 1***.
- Point out the reference point on the turntable with the robot's TCP.
- Modify the position by pressing the function key ***ModPos***.
- Rotate the second axis to a greater angle and repeat the above for the points ***Point 2*** to ***Point n***.
- Press ***OK*** to calculate the frame of the second axis.

To confirm/cancel the new axis definition

When ***OK*** is pressed after the points have been modified for an axis, a dialog like the one in Figure 22 will appear.

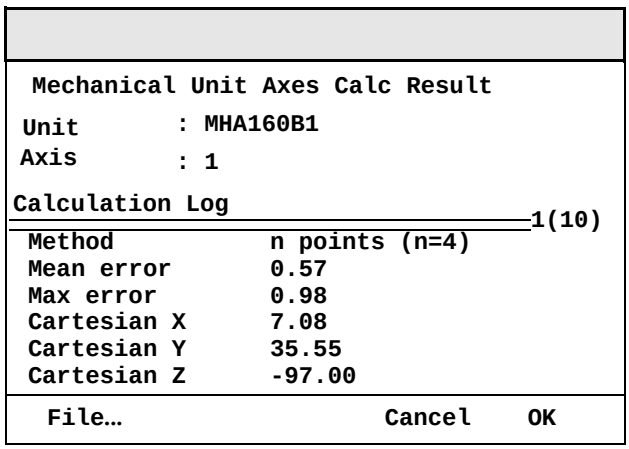


Figure 22 The result of the first axis definition.

The calculation log shows the calculated frame expressed in the world coordinate system.

Field	Description
Unit	The name of the mechanical unit for which the definition of the axis is to be done.
Axis	The chosen axis.

List contents	Description
Method	Displays the selected method.
Mean error	The accuracy of the robot positioning relative to the reference point.
Max error	The maximum error for one positioning.
Cartesian X	The x coordinate for the frame.
Cartesian Y	The y coordinate for the frame.
Cartesian Z	The z coordinate for the frame.
Quaternion 1-4	Orientation components for the frame.

The calculation result can be saved in a separate file for later use in a PC:

- Press the function key ***File***.
- Specify a name and a location where to save the result.
- Choose ***OK*** to confirm the save.

If the estimated error is

- acceptable, press ***OK*** to confirm the new axis definition. Now the next axis can be defined if necessary.
- not acceptable, redefine by pressing ***Cancel***.

• Choose **File: Restart** in the *Service* window to activate the user frame.

The user frame definition is now completed, but before proceeding with other tasks, verify it by jogging the mechanical unit in coordinated mode.

Note The user frame is stored in the system parameters as the base frame of the external mechanical unit. The user frame in the corresponding work object is therefore not used.

7 Defining Tools

The position of the robot and its movements are always related to its tool coordinate system, i.e. the TCP and tool orientation (see Figure 23). To get the best performance, it is important to define the tool coordinate system as correctly as possible. For more information, see the RAPID Reference Manual/ Motion and I/O Principles.

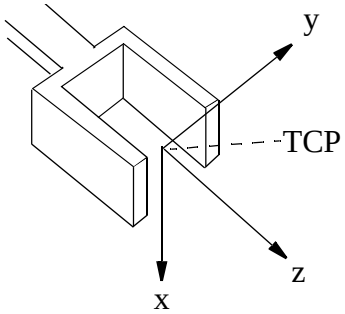


Figure 23 The tool coordinate system for a gripper.

A tool coordinate system can either be defined manually or the robot can be used as the measuring tool. Manual definitions can be used if accurate data for the dimensions of the tool is available or if minor corrections are to be done.

7.1 Creating a new tool

A tool should normally be placed in the system module, *User*. In that way, it will be common to all programs, which means that if a TCP is modified, all programs will automatically be affected. The tool can then also be used for jogging when there is no program in the program memory.

• Open the *Program Data Types* window by choosing **View: Data Types**.

• Select the type *tooldata* and press Enter [↵].

• Create the new tool using one of the following alternatives:

- **alt 1**. Press the function key ***New***.
  The tool's TCP and orientation will then be the same as the robot's mounting flange.
- **alt 2**. Select an existing tool and press the function key ***Dupl***.
  The tool's TCP and orientation will then be the same as the one duplicated.

A window appears, displaying the name of the data.

• If you want to change the name, press Enter [↵] and specify a new name.

• Press the function key ***Decl***.

A dialog box appears, displaying the basic tooldata declaration.

• If you want to save the data in another module, select the field ***In Module*** and press Enter [↵]. Specify the name of the module in which the data is to be saved.

• Press ***OK*** to confirm.

Note: Do not change the type of the tool. This must always be of the persistent type.

7.2 Manually updating the TCP and weight of a tool

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *tooldata* and press Enter ↵ .
- Select the tool to be changed and press Enter ↵ .
- Select the TCP component (x, y, z) that you wish to change.
- Change the value using the numeric keyboard. To enter a decimal point (.) or minus sign (-), use the function keys.
- Select the *mass* component.
- Change the weight using the numeric keyboard.
- If the tool is stationary, i.e. not mounted on the robot, change the component *robhold* to FALSE. For more information about stationary tools see *Stationary tool* on page 33.
- Choose ***OK*** to confirm the change.

Note: Only the mass of the tool should be specified. A payload handled by a gripper is specified by the instruction GripLoad.

7.3 Methods of defining the tool coordinate system

To define the TCP of a tool, you need a world fixed tip within the robot's working space. You then jog to (at least) four robot positions with different orientations, as close as possible to the world fixed tip (see Figure 24). These positions are called *approach points*.

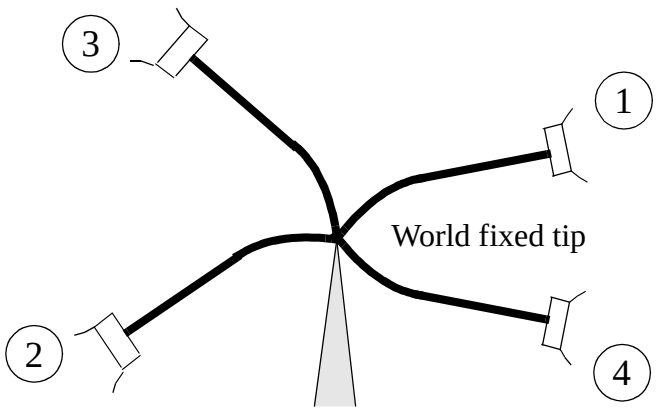


Figure 24 Approach points for a tool's TCP.

To define a complete orientation of a tool, you move any position on the desired z axis and any position on the desired x axis to the world fixed tip. These positions are called *elongator points* (see Figure 25). This can be done by fitting an elongator to the tool to define the z and x directions or by aligning the tool according to the world coordinate system and then jogging the robot in these directions.

Note The elongator points must be defined with the same orientation as the last approach point used.

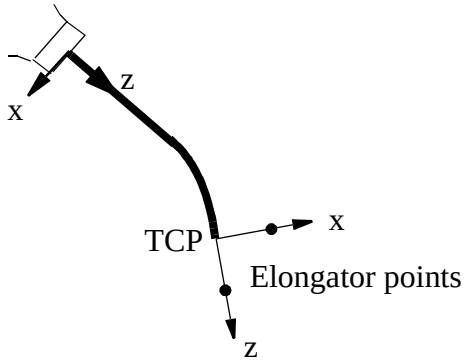


Figure 25 Elongator points for a tool's orientation.

If you only want to define the TCP, only the world fixed tip is needed. If you only need a definition of the orientation in the z direction, the elongator will only point to z.

The following methods are supported:

- ***4-point TCP***
  Four approach points are used to define the TCP. The orientation will be set according to the wrist coordinate system (see Figure 26).

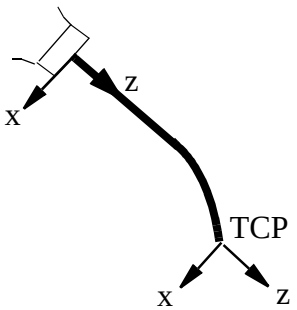


Figure 26 Using the 4-point method, only the TCP is defined. The tool direction will correspond to the wrist coordinate system.

- ***4-p TCP ORIENT NOT SET***
  The same as 4-point TCP but the orientation will not be changed.

- ***5-point TCP&Z***
  Four approach points are used to define the TCP and one elongator point is used to define the z direction of the tool. The x and y directions will be as close as possible to the corresponding axes in the wrist coordinate system (see Figure 27).

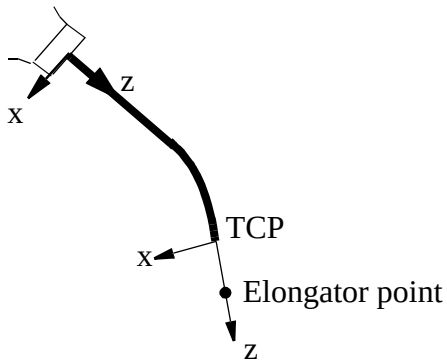


Figure 27 Using the 5-point method, the TCP and the tool's z direction are defined. The x and y directions are set automatically by the robot.

- ***6-point TCP&ZX***
  Four approach points are used to define the TCP, one elongator point is used to define the z direction and one elongator point is used to define the x direction of the tool (see Figure 28).

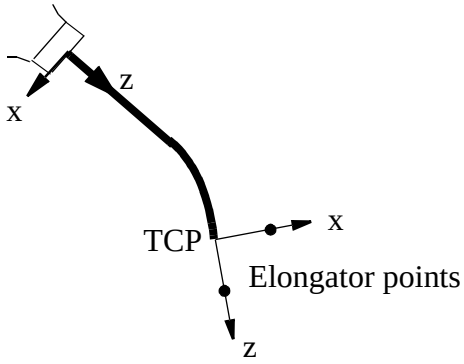


Figure 28 Using the 6-point method, the TCP and all the tool's directions are defined.

7.4 Using the robot to change the TCP and orientation of a tool

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *tooldata* and press Enter ↵.
- Select a tool (or create a new tool, see ***Creating a new tool*** on page 28).
- Choose **Special: Define Coord**.

A dialog box appears, displaying the points defined by whichever method was used (see Figure 29).

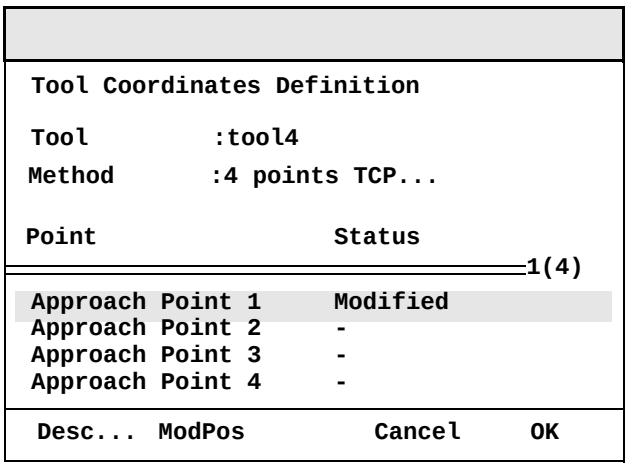


Figure 29 The robot can be used to define the tool coordinate system.

The status can be defined as follows:

Status	Meaning
-	No position defined
Modified	Position modified

To choose a definition method

Before you start modifying any positions, make sure the desired method is displayed. See ***Methods of defining the tool coordinate system*** on page 29.

- Select the field ***Method*** and press Enter ↵.
- Choose a method and press ***OK***.

To record Approach Points

- Select the first point ***Approach Point 1***.
- Jog the robot as close as possible to the world fixed tip.
- Modify the position by pressing the function key ***ModPos***.
- Repeat the above for the points ***Approach Point 2-4***.

To record Elongator Point Z (if the 4-point TCP method is not used)

- Select ***Elongator z Point***.
- Jog - without changing the orientation from the last approach point - any point on the desired positive z axis to the world fixed tip. An extension should be fitted to obtain better accuracy.
- Modify the position by pressing the function key ***ModPos***.

To record Elongator Point X (only if the 6-Point TCP&XZ method is used)

- Select ***Elongator x Point***.
- Jog - without changing the orientation from the last approach point - any point on the desired positive x axis to the world fixed tip.
- Modify the position by pressing the function key ***ModPos***.

To calculate the tool coordinate system

- Press ***OK*** to calculate the tool coordinate system.

When the calculation is finished, a dialog like the one in Figure 30 will appear.

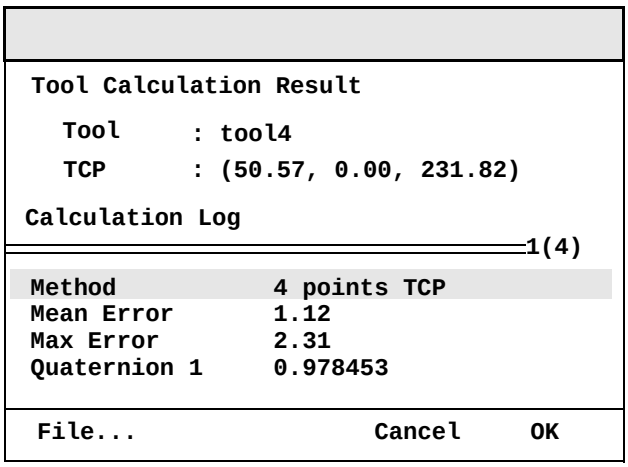


Figure 30 The result of a tool calculation.

Field	Description
TCP	The values of the calculated TCP.
Mean Error	The average distance that the approach points are from the calculated TCP, i.e. how accurately the robot was positioned relative to the tip.
Max Error	The maximum error for one approach point.

The calculation result can be saved in a separate file for later use in a PC. However, this file cannot be read by the robot:

- Press the function key ***File***.
- Specify a name and a place to save the result.
- Choose ***OK*** to confirm the save.

• If the estimated error is

- acceptable, press ***OK*** to confirm the new tool coordinate system;
- not acceptable, redefine by pressing ***Cancel***.

The definition is now complete, but before proceeding with other tasks, verify it by linearly jogging in the tool coordinate system and by reorienting the TCP.

If the tool has been stored in a system module, save this module.

7.5 Stationary tool

When using a stationary tool, the robot is holding the work piece and the tool is stationary in the room. In this case the TCP coordinates are related to the world coordinate system, and the work object (i.e. the user coordinate system) is related to the wrist coordinate system.

Creating a new tool.

• The tool is created as described in previous chapters.

• The component *robhold* is changed to FALSE.

Creating a corresponding work object

When using a stationary tool, it is also necessary to use a work object held by the robot.

• The work object is created as described in *Creating a new work object* on page 36.

• The component *robhold* is changed to TRUE.

Methods for defining the tool coordinate system

The methods are the same as for a TCP mounted on the robot. However in this case, the reference tip is mounted on the robot and the robot is moved, so as to bring the tip to the stationary tool TCP. The tip must be defined and activated as a tool before the definition of the stationary tool may be done.

• Define and activate the tool, which should be used as a pointing tip, and which is mounted on the robot.

• Now the same methods for defining the stationary tool may be used, as described in *Manually updating the TCP and weight of a tool* on page 29 and *Using the robot to change the TCP and orientation of a tool* on page 31. Use the robot mounted tip to point out the stationary TCP with four approach points, and if needed, the z and x directions of the axes. It is possible to use the same positioning for all four TCP approach points to perform a faster frame definition. However, it is recommended to point out the stationary TCP with different orientations to obtain a reliable statistical result. The point that is used to approach the stationary TCP must be the active TCP (hold by the robot).

Note: If the stationary tool is to be used with coordinated track motion , the coordination must be active during the calibration of the stationary tool.

8 Work Objects and Program Displacements

8.1 General

All programmed positions are related to a program displacement frame, which in turn is related to the object frame, related to the user frame, related to the world frame. Both object and user frames are included in a work object, which may be added to each move instruction. See Figure 31.

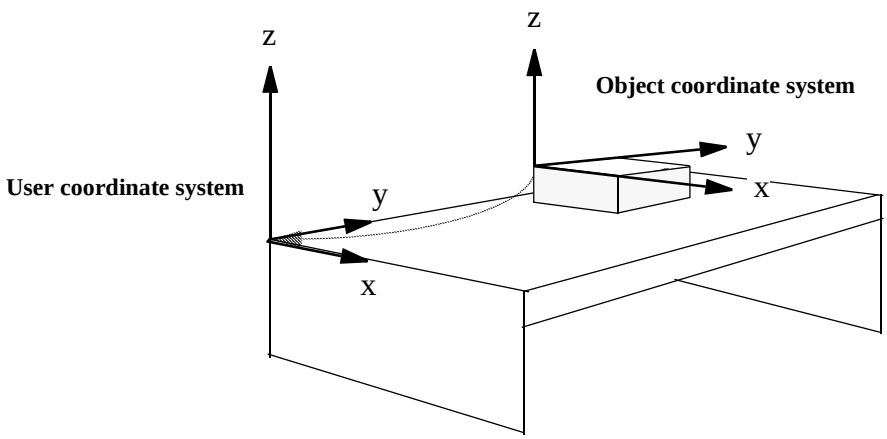


Figure 31 A user and an object coordinate system describe the position of a work object.

The intention is to use the work object to define both the position of a table (user frame) and the position of the object to work on (object frame). When the table or the object is moved, the program may still work if the corresponding work object is updated. These coordinate systems are very well suited to off-line programming since the positions specified can usually be taken directly from a drawing of the work object.

The program displacement coordinate system is used for small temporary displacements, e.g. as the result of a search operation. This displacement is modal, i.e. it is activated in a separate instruction and then it remains active until it is deactivated in another separate instruction. See Figure 32.

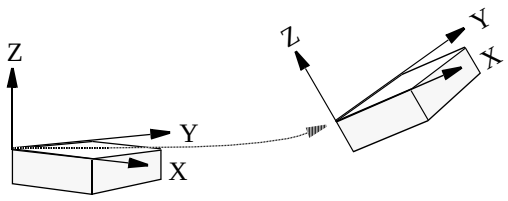


Figure 32 Using a displacement frame, all positions in the program can be displaced.

All such program displacements include both robot displacements and external axes displacements.

Please note the difference between work object and program displacement. The work object used must be added to each move instruction and it must be active when programming the move instruction. It should be included from the beginning because it is a little tricky to add it afterwards. A program displacement, however, which is activated in a separate instruction, is very easy to add afterwards.

8.2 Using work objects

In the checklist below, the steps required to define and use a work object are described. In each step, there may be a reference to another chapter in this manual, where more details of the specific actions to be taken will be found.

- Before starting to program, the work objects to be used must be defined. First create a new work object and give it a name, e.g. “wobj1”, see *Creating a new work object* on page 36.
- Define the work object by using the robot to point out three points on the user frame and the object frame respectively. See *Using the robot to change the work object* on page 38. Please note that if the same positions are used both for the user frame and for the object frame, then all the locations will go into the user frame and the object frame will still be zero. It should also be noted that it is possible to update the values of the work object manually. See *Manually updating the user and object coordinate system of the work object* on page 37.
- Now check that the definition of the work object is correct by jogging the robot in the object coordinate system. This may be done by choosing the ***Wobj*** in the field ***Coord*** in the jogging window, and the work object, e.g *wobj1*, in the field ***Wobj***, and then jogging the robot.
- When programming it is important to have the work object, in this case *wobj1*, programmed as an argument in each move instruction. This will be automatically added to the move instruction, if the work object is activated in the jogging window before starting the programming.

8.3 Creating a new work object

A work object should normally be placed in the system module, *User*. In this way it will be common to all programs, which means that if a work object is modified, all programs will also automatically be modified. The work object can also be used for jogging when there is no program in the program memory.

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *wobjdata* and press Enter [↵].
- Create the new work object using one of the following alternatives:
  - **alt 1**. Press the function key ***New***.
    The user and object coordinate systems will then coincide with the world coordinate system.
  - **alt 2**. Select an existing work object and press the function key ***Dupl***.
    The coordinate systems will then be the same as those duplicated.

A window appears, displaying the name of the data.

- If you want to change the name, press Enter [↵] and specify a new name.
- Press the function key ***Decl***.

A dialog box appears, displaying the basic wobjdata declaration.

- If you want to save the data in another module, select the field ***In Module*** and press Enter ↵ . State the name of the module where the data is to be sent.
- Press ***OK*** to confirm.

Note: Do not change the work object type. This must always be of the persistent type.

8.4 Manually updating the user and object coordinate system of the work object

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *wobjdata* and press Enter ↵ .
- Select the work object to be changed and press Enter ↵ .
- Select the component (x, y, z, q1-q4) that you wish to change.
- Change the value using the numeric keyboard. To enter a decimal point (.) and minus sign (-), use the function keys.
- Choose ***OK*** to confirm.

Note If the work object is defined using a movable user coordinate system, only the object coordinate system need be defined. The user coordinate system is defined in the Service window. See *Coordinated external axes* on page 16.

8.5 Methods of defining a work object

The methods used to define the user and object coordinate system are called:

- ***No change***
  No changes to the definition of the user or object coordinate system will be made, i.e. the definition of the user or object frame will be left as it is.
- ***3-point***
  Three points are used: two points on the x axis and one point on the y axis (see Figure 33). A tool with a known TCP is required.

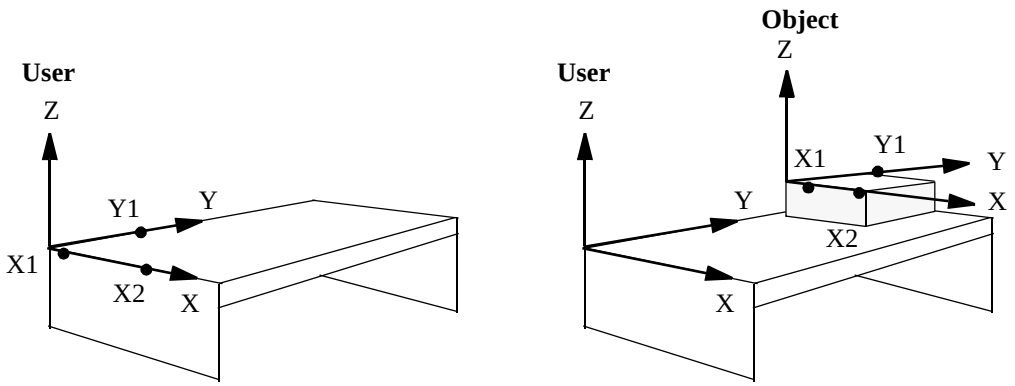


Figure 33 Measuring points for defining a work object.

8.6 Using the robot to change the work object

- Choose **View: Data Types**.
- Select the type *wobjdata* and press Enter ↵.
- Select the work object to be defined (or create a new one, see *Creating a new work object* on page 36).
- Choose **Special: Define Coord**.

A dialog box appears, displaying the points defined by the method that was used (see Figure 34).

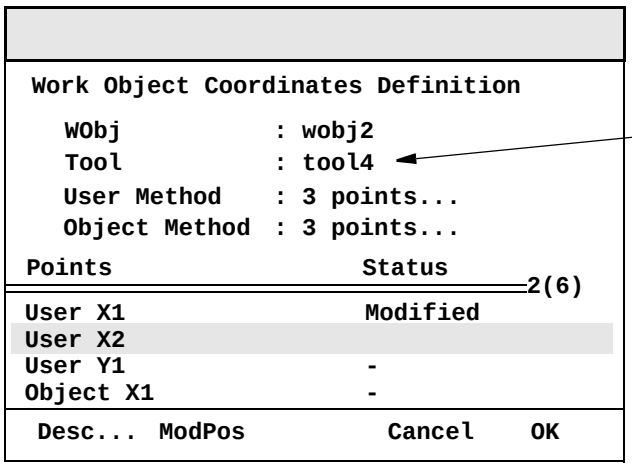


Figure 34 The robot can be used to define the position of the work object.

The status can be defined as follows:

Status	Meaning
-	No position defined
Modified	Position modified

To record Measuring Points for the user coordinate system

Note If the work object is defined using a movable user coordinate system, the user coordinate system is defined in the Service window. See *Coordinated external axes* on page 16.

- Select the first measuring point ***User X1***.
- Jog the robot as close as possible to a point on the x axis.
- Modify the position by pressing the function key ***ModPos***.
- Select the measuring point ***User X2***.
- Jog the robot as close as possible to a point on the x axis defining the positive x direction.
- Modify the position by pressing the function key ***ModPos***.

• Select the measuring point ***User Y1***.

• Jog the robot as close as possible to a point on the positive y axis.

• Modify the position by pressing the function key ***ModPos***.

To record measuring Points for the object coordinate system

• Select the first measuring point ***Object X1***.

• Jog the robot as close as possible to a point on the x axis.

• Modify the position by pressing the function key ***ModPos***.

• Select the measuring point ***Object X2***.

• Jog the robot as close as possible to a point on the x axis defining the positive x direction.

• Modify the position by pressing the function key ***ModPos***.

• Select the measuring point ***Object Y1***.

• Jog the robot as close as possible to a point on the positive y axis.

• Modify the position by pressing the function key ***ModPos***.

To calculate the user and object coordinate system

• Press ***OK*** to calculate the coordinate systems.

When the calculation is finished, a dialog like the one shown in Figure 35 will appear.

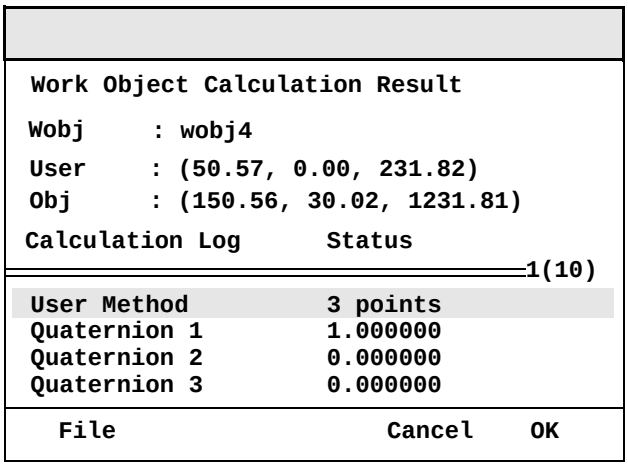


Figure 35 The result of a work object calculation.

Field	Description
User	The origin of the user coordinate system.
Obj	The origin of the object coordinate system.

The calculation result can be saved in a separate file for later use in a PC.
Note, however, that this file cannot be read by the robot:

• Press the function key ***File***.

• Specify a name and a place to save the result.

• Choose ***OK*** to confirm the save.

The definition is now complete, press ***OK*** to confirm the new work object, but before proceeding with other tasks, verify it by jogging linearly in the work object's coordinate system.

If the work object was stored in a system module, save this module.

8.7 Defining a moveable object frame

Method 1

• Use the method for defining a work object. See *Using the robot to change the work object* on page 38. When using this method, please observe that the coordination flag, i.e. the component *ufprog* in the work object data must be temporarily set to TRUE. You must point out three positions for the user system (which must be placed as the coordinated one) and three positions for the object system.

If the user system is not possible to reach, use method 2 or 3 below.

Method 2

• Activate the coordinated work object and jog the robot to the point where you want to place the origin of the object frame.

• Read the coordinates, x, y, z for this position in the jogging window.

• Write these values in the *o_frame* component of the work object data.

This will shift the object frame to the new position, with the same orientation as the user frame. If you want another orientation, use method 3.

Method 3

• Activate the coordinated work object (suppose it is named *co_wobj*), create three positions, e.g. *p1*, *p2* and *p3*. *p1* should be located at the origin of the shifted object frame, *p2* on the x axis and *p3* in the x-y plane.

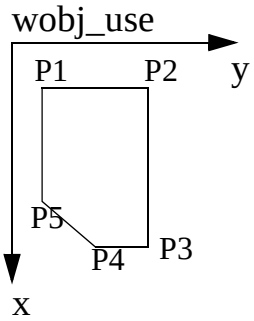
• Program and execute the instruction
co_wobj.oframe: = DefFrame(p1, p2, p3);

8.8 How to use different work objects to get different displacements

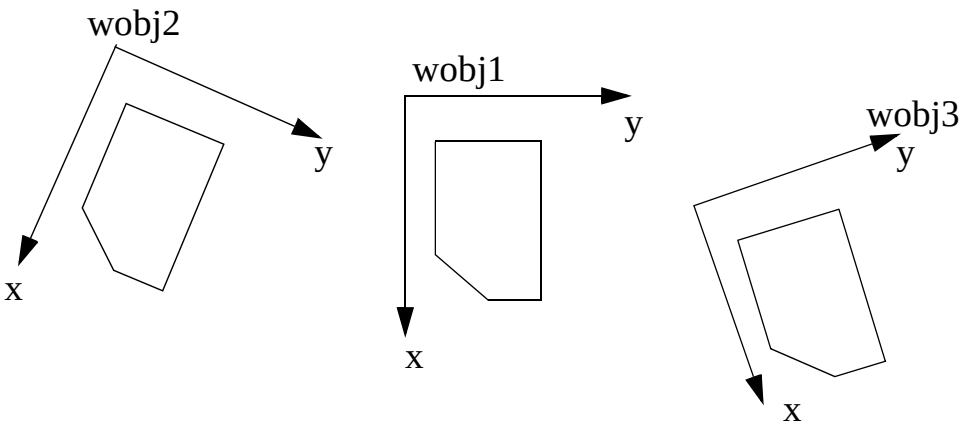
Suppose you have used the work object *wobj_use* when creating a procedure, *draw_fig*, as below.

```
MoveL p1, v200, z1, tool1\WObj:=wobj_use;
MoveL p2, v200, z1, tool1\WObj:=wobj_use;
MoveL p3, v200, z1, tool1\WObj:=wobj_use;
```

```
MoveL p4, v200, z1, tool1\WObj:=wobj_use;
MoveL p5, v200, z1, tool1\WObj:=wobj_use;
```



Now you want it to be performed displaced, corresponding to *wobj1*, *wobj2* or *wobj3*, see below.



Suppose that the value of *reg1* is used to control which work object should be used.

If *reg1 = 1*, *wobj1* should be used; if *reg1 = 2*, *wobj2* should be used; and if *reg1 = 3*, *wobj3* should be used.

The program below will set wobj_use = wobj1 if reg1 = 1, then call the *draw_fig* procedure, etc.

```
IF reg1=1 THEN
                    wobj_use:=wobj1;
                    draw_fig;
ENDIF
IF reg1=2 THEN
                    wobj_use:=wobj2;
                    draw_fig;
ENDIF
IF reg1=31 THEN
                    wobj_use:=wobj3;
                    draw_fig;
ENDIF
```

8.9 How to adjust the program vertically using the object frame

When running your program in the location defined by *wobj2*, suppose you find it is positioned a little too high. The vertical position can be adjusted by moving the object coordinate system a small amount vertically, relative to the user coordinate system, i.e. the z coordinate for object is changed. E.g. if the robot is to work a little lower, then the z value should be decreased.

8.10 Using program displacement

A program displacement is set with a *pose* data, using a *PDispSet* instruction. This will store the program displacement in a system variable, *C_PROGDISP*, holding also displacement values for external axes. The current value in *C_PROGDISP* is used in all movement instructions and added to the programmed positions. The program displacement is cleared, when a *PDispOff* instruction is executed, resulting in no further displacement.

A *PDispOn* instruction will both calculate a new program displacement, from the difference between two positions, and store this displacement in the *C_PROGDISP* variable. When this instruction has been executed a new program displacement will become active.

The following example will illustrate how to use a *PDispOn* instruction in combination with a *SearchL* instruction, to make a movement on different locations, depending on the search point.

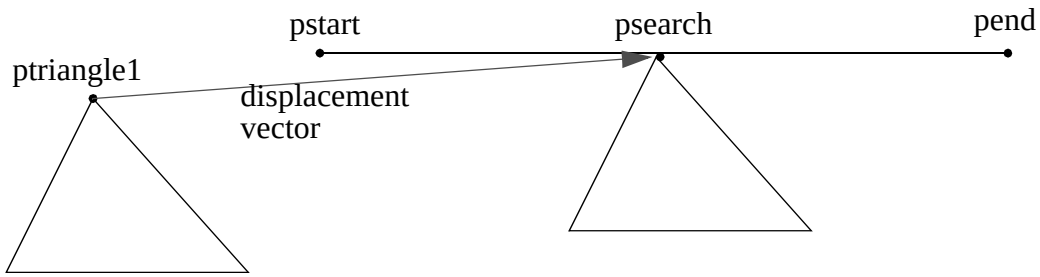
The program should do the following:

- Go to a start point, *pstart*, for searching.
- Make a linear search from the start position to an end position, *pend*. When a digital input *di1* is set, the robot should stop the movement and draw a figure, *triangle*, the position of which will depend on the search point, *psearch*.

The figure, *triangle*, is programmed with no displacement active and with the first position in *ptriangle1*.

The program may look like:

```
MoveL pstart, v200, fine, tool1;
SearchL \Stop, di1, psearch, pend, v100, tool1;
PDispOn \ExeP: = psearch, ptriangle1, tool1;
triangle;
PDispOff
etc.
```



8.11 Creating a new displacement frame

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *pose* and press Enter ↵.
- Create the new displacement frame using one of the following alternatives:
  - **alt 1**. Press the function key ***New***.
    The displacement frame will then have no translation or rotation.
  - **alt 2**. Select an existing displacement frame and press the function key ***Dupl***.
    The displacement frame will then be the same as the one duplicated.

A window appears, displaying the name of the data.

- If you want to change the name, press Enter ↵ and specify a new name.
- Press ***OK*** to confirm.

8.12 Manually updating a displacement frame

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *pose* and press Enter ↵.
- Select the displacement to be changed and press Enter ↵.
- Select the frame component (x, y, z, q1-q4) that you wish to change.
- Change the value using the numeric keyboard. To enter a decimal point (.) and minus (-), use the function keys.
- Choose ***OK*** to confirm the change.

8.13 Methods for defining a displacement frame

The following method is supported:

- ***n-point***
  At least three well-defined points on an object at its initial position and the same points when the object is in its new position (see Figure 36) are used to define the displacement frame.

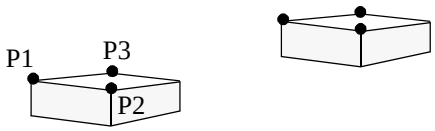


Figure 36 A displacement frame can be defined by moving the robot to a number of points.

8.14 Using the robot to change a displacement frame

- Open the *Program Data Types* window by choosing **View: Data Types**.
- Select the type *pose* and press Enter ↵.
- Select the displacement frame to be defined (or create a new one, see ***Creating a new displacement frame*** *on page 43*).
- Choose **Special: Define Coord**.

A dialog box appears, displaying the points defined by the method that was used (see Figure 37).

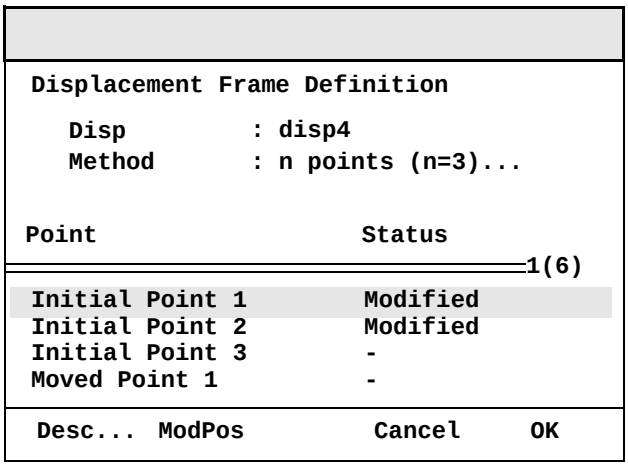


Figure 37 Displacement frame definition dialog

The status can be defined as follows:

Status	Meaning
-	No position defined
Modified	Position modified

To choose the definition method

Before you start modifying any positions, make sure the ***n-point*** method, together with the number of points that you want to use, is displayed:

- Select the field ***Method*** and press Enter ↵.
- Enter the desired number of points and press ***OK***.

To record the Initial Points

- Select the first definition point ***Initial Point 1***.
- Jog the robot as close as possible to a well-defined position on the object.
- Modify the position by pressing the function key ***ModPos***.
- Repeat the above for the points ***Initial Point 2***, ***Initial Point 3***, etc.

To record Moved Points

- Move the object to its new position.
- Select the first definition point ***Moved Point 1***.
- Jog the robot as close as possible to the same position on the object as for ***Initial Point 1***.
- Modify the position by pressing the function key ***ModPos***.
- Repeat the above for the points ***Moved Point 2***, ***Moved Point 3,*** etc.

To calculate the displacement frame

- Press ***OK*** to calculate the displacement frame.

When the calculation is finished, a dialog like the one shown in Figure 38 will appear.

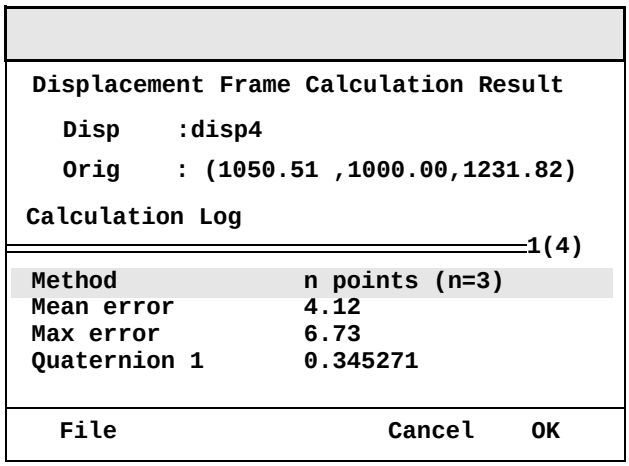


Figure 38 The result after a displacement frame calculation.

Field	Description
Orig	The origin of the displacement frame.
Mean Error	The average distance that the points are from the original points, i.e. how accurately the robot was positioned.
Max Error	The maximum error for one point.

The calculation result can be saved in a separate file for later use in a PC. Note, however, that this file cannot be read by the robot:

- Press the function key ***File***.
- Specify a name and a place to save the result.
- Choose ***OK*** to confirm the save.

• If the estimated error is

- acceptable, press ***OK*** to confirm the new displacement frame;
- not acceptable, redefine by pressing ***Cancel***.

CONTENTS

Page

1 The Production Window **3**
2 Reading a Program **4**
3 Changing the Override Speed **5**
4 Changing the Program Running Mode **5**
5 Starting the Program **6**
5.1 Restarting after a stop 7
5.2 Starting a program from the beginning 7
6 Stopping the Program **7**
7 Tuning position **8**
8 Operator Dialogs **9**

Production Running

The Production window appears automatically on the teach pendant display when the power is switched on, and the operating mode selector is in the Auto position. You can also call it up by pressing and choosing ***Production***.

1 The Production Window

The Production window is used to start and stop program execution (see Figure 1).

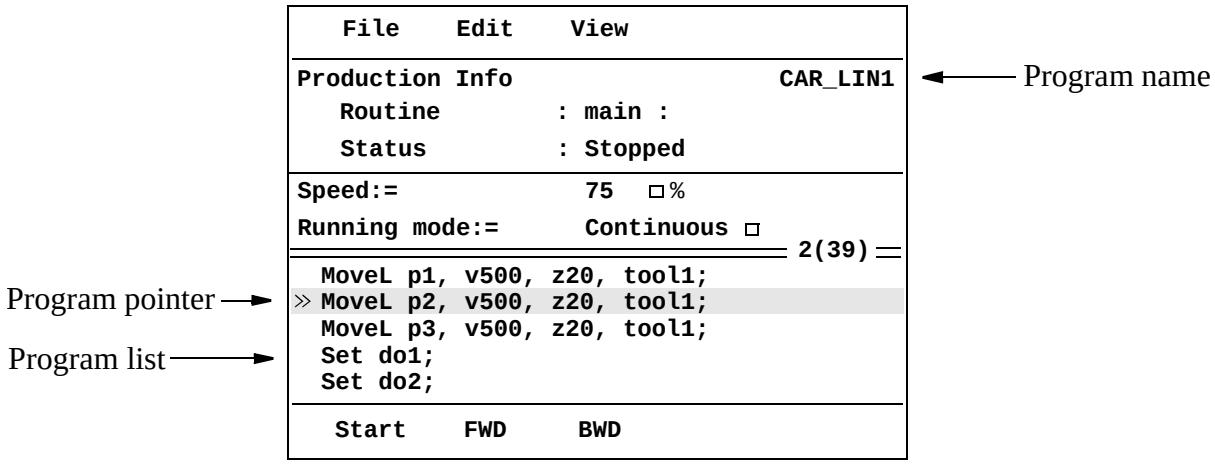


Figure 1 All production runs are controlled from the Production window.

Before starting the program, check the program name to see that it is the correct program. The Program name is displayed in the right hand upper corner of the window.

• Choose **View:Info** to open the window *Production Info*.

To start the program, see *Starting the Program* on page 6.

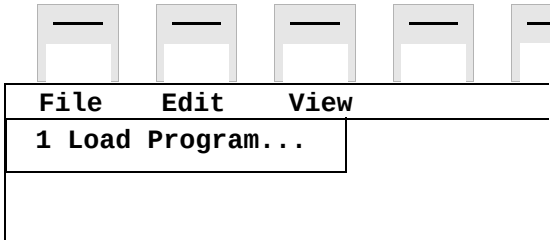
If the Status field indicates NOT LOADED, then you must load a program (see *Reading a Program* on page 4).

Field:	Indicates:		
Routine	The subprogram that is being run		
Status	NOT LOADED	=	no program is loaded
	STOPPED	=	a program is loaded and it can be executed (PP is set)
	RUNNING	=	program execution is in progress
	NOT EXECUTABLE	=	a program is loaded but cannot be executed
Speed	The chosen speed correction as a percentage		
Running mode	Continuous	=	continuous execution
	Cycle	=	the program is executed once
Program list	The part of the program that is being run		
Program pointer	The instruction to be executed when ***Start*** is pressed.		

2 Reading a Program

A program can be read from a diskette or from the robot's mass memory. To open a program, do as follows:

• Choose **File: Load Program**.



The following dialog box will appear (see Figure 2).

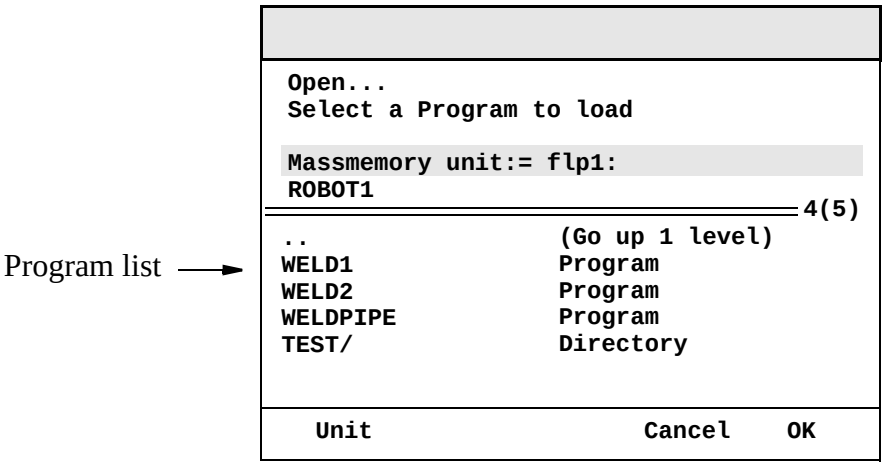


Figure 2 The dialog box displays a list of all available programs.

The Mass memory unit field indicates:

- **flp1** to denote a diskette
- **ram1disk** to denote the robot's internal memory (the RAM disk)

• Press ***Unit*** until the desired unit is displayed.

• Choose the desired program – use ArrowUp or ArrowDown to scroll through the list: select . . to go up one level and press to go down one level.

• Press ***OK***.

3 Changing the Override Speed

The speed of the robot can be adjusted while running production. The function keys indicate how the speed can be decreased or increased.

–%	Decreases the value by 5% (or 1% if <5%)
+%	Increases the value by 5% (or 1% if <5%)
25%	Sets the value to 25%
100%	Sets the value to 100%

To override the speed, do as follows:

• Select the middle part of the display by pressing .

• Using one of the arrow keys, select the field for the corrected speed (see Figure 3).

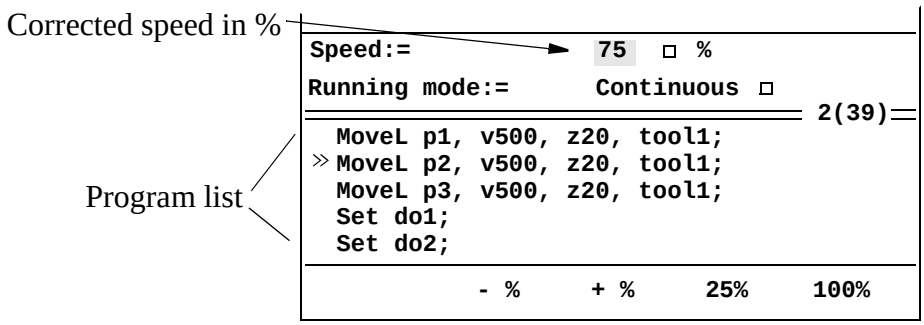


Figure 3 The function keys can be used to both increase and decrease the programmed speed.

• Press the desired alternative.

• To return to the program list, use .

4 Changing the Program Running Mode

A program can be run in either of the following two ways:

- ***Cont*** – continuous execution
- ***Cycle*** – the program is executed once.

You can change the program running mode in the ***Running mode*** field:

• Select the middle part of the display by pressing .

• Select ***Running mode*** (see Figure 4).

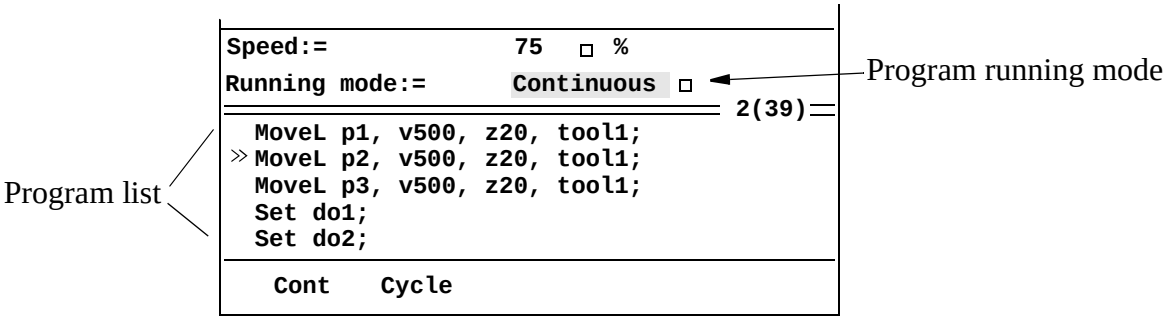


Figure 4 The function keys are used to select the different program running modes.

- Press the desired function key ***Cont*** or ***Cycle***.
- To return to the program list part, use .

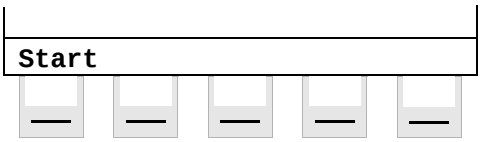
5 Starting the Program

When the program is started, the robot and any peripheral equipment will start to move. Check that all preparations have been made for program execution. Make sure that the workcell is clear of all personnel before starting the robot.

If NOT LOADED is displayed on the program status line, then a program must be loaded (see *Reading a Program* on page 4).

If a program is loaded and is executable, STOPPED will be displayed on the program status line and the program can be started:

- Press the function key ***Start***.



When a program is executing the current executing instruction is shown in the window (see Figure 5).

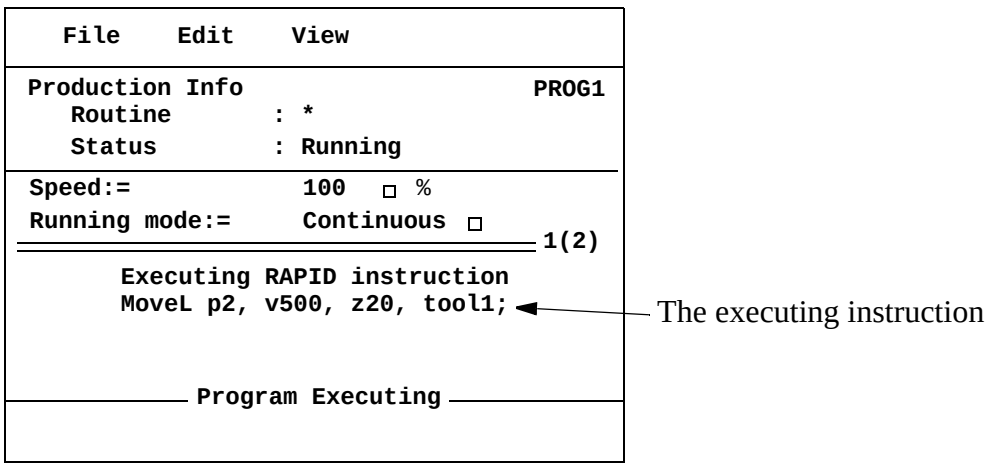


Figure 5 The current instruction is shown while the program is running.

5.1 Restarting after a stop

If you wish to restart program execution from where it was interrupted:

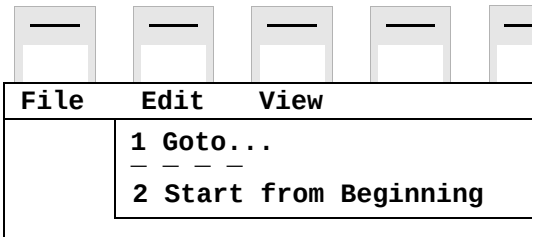
• Press ***Start***.

The program can also be restarted from the beginning. This is described below.

5.2 Starting a program from the beginning

To start again from the beginning, proceed as follows:

• Choose **Edit: Start from Beginning**.



• Press ***OK*** to confirm.

The program pointer ≫ will then move to the first instruction in the program.

• Press ***Start***.

6 Stopping the Program

Program execution can be stopped by pressing the stop button on the teach pendant (see Figure 6).

In case of an emergency, press one of the emergency stop buttons instead. This will cut off the power supply to the robot motors and engage all brakes.

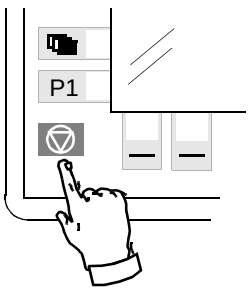
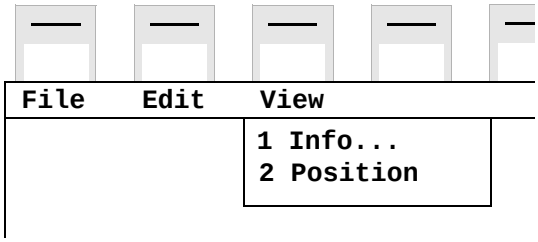


Figure 6 This stop button is used to stop the program.

7 Tuning position

• Choose **View: Position**



The tuning function in the Production window makes it possible to tune the x, y and z coordinates of a robot position (see Figure 7). The tuning can be performed either when status is Stopped or when status is Running.

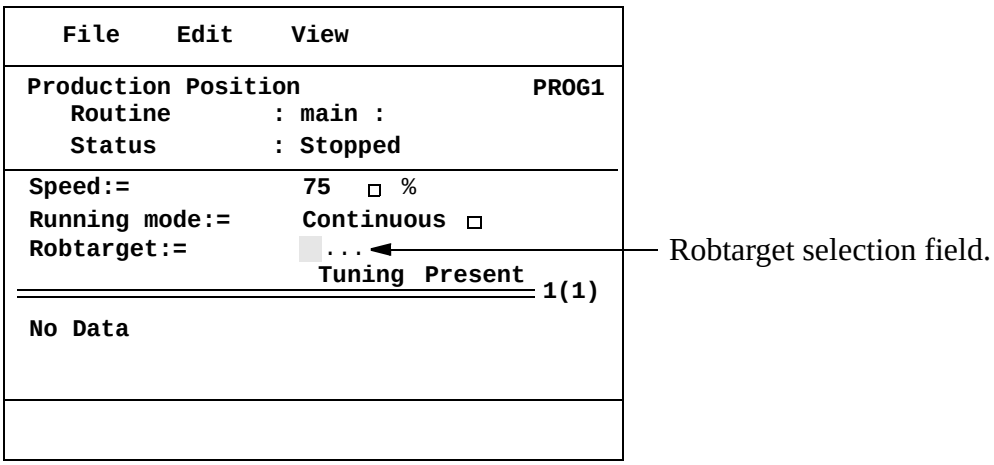


Figure 7 The Production Info view. No robtarget selected.

• Select the field ***Robtarget*** and press Enter ↵.

• Choose the position to be tuned in the list that will appear.

• Press ***OK*** or Enter ↵ to confirm the choice.

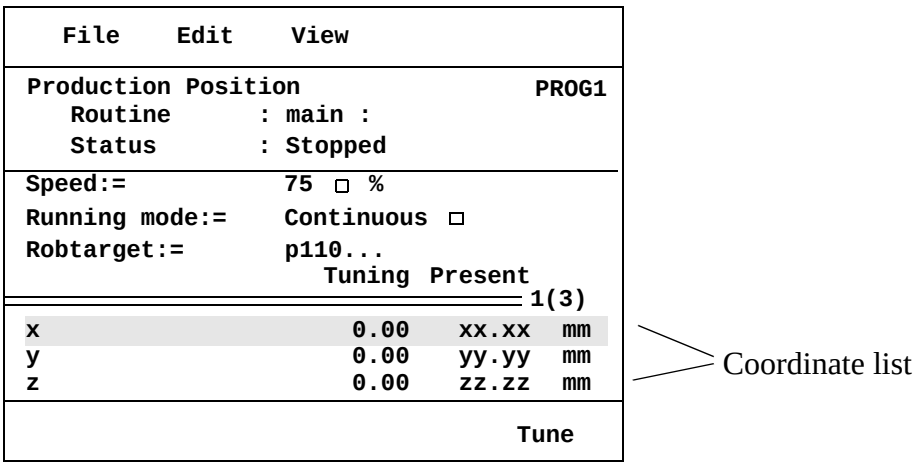


Figure 8 The Production Position view with a robtarget selected.

• Choose the x, y or z coordinate in the coordinate list (see Figure 8).

• Press: **Tune**

A dialog will appear where you can tune the position.

• Enter the desired tuning value and press Enter ↵.

- No change = 0.
- Max. change in one step = ±10 mm

Several steps can be entered. The position data is changed immediately after each step but will not affect the robot path until the next instruction using this position data is executed. The values in the ***Present*** column will be used in this instruction.

The total tuning will be displayed in the ***Tuning*** column.

Note If a named position data is modified, all instructions which refer to that position data will be affected. Unnamed positions (marked as * in the instruction) cannot be tuned.

See also Chapter 8 Programming and Testing - *Tuning position during program execution.*

The tuning function can be disabled in automatic mode. See chapter 12 System Parameters - *Topic:Teach Pendant.*

8 Operator Dialogs

Special instructions can be created in the program and used as a form of communication between the program and the operator (see Figure 9).

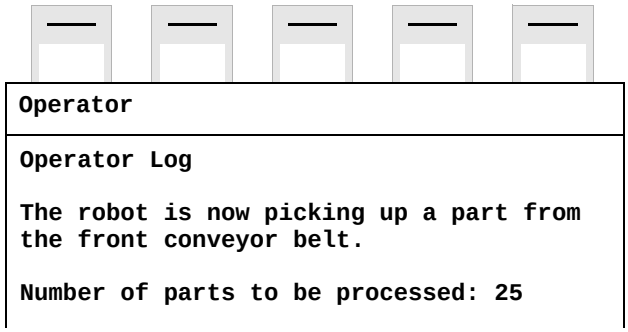


Figure 9 Example of a message sent to the operator.

• You can return to the *Production* window by pressing ⧉ and choosing ***Production***.

Sometimes, the operator must respond before program execution can continue (see the example in Figure 10).

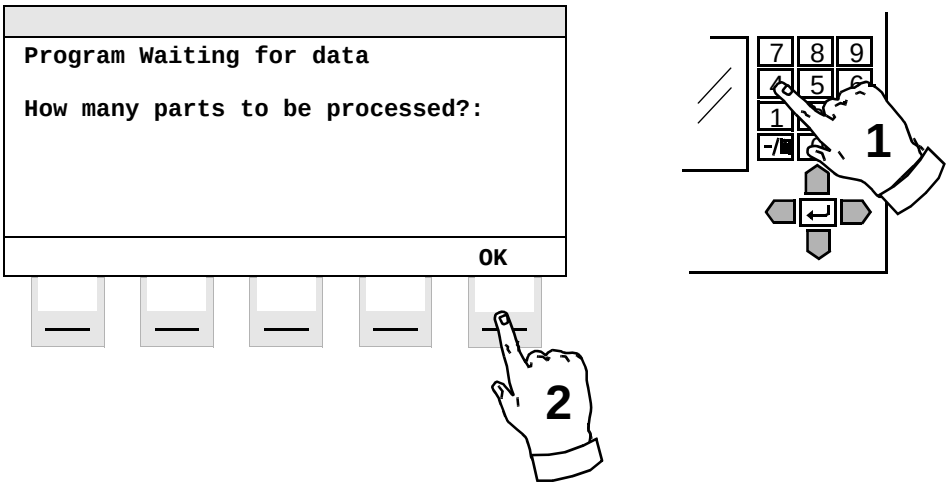


Figure 10 Use the numeric keyboard to answer questions from the program.

- Use the numeric keys when the reply is a numeric value.
- Press ***OK***.
- If text is displayed above the function keys, you can give your answer by pressing the desired alternative (see Figure 11).

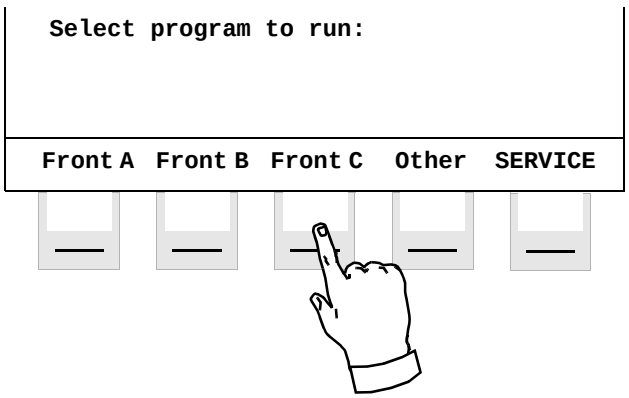


Figure 11 Operator dialogs can be tailor-made to suit any robot installation.

The dialog window shown in Fig. 10 can only be exited in one of two ways, either by answering the question or by stopping program execution.

CONTENTS

Page

System Parameters 3
1 Changing a Parameter 3
1.1 Subdivision of parameters 3
1.2 Changing a parameter 3
1.3 Deleting a parameter 4
1.4 Generating a restart 4
1.5 Viewing the last changes that were made 5
1.6 Checking Parameters 5
2 Saving and Loading Parameters 6
2.1 Saving parameters to diskette or some other mass storage device 6
2.2 Loading parameters from a diskette or some other mass storage device 7
3 Topic: IO Signals 9
3.1 Defining I/O Units 9
3.2 Additional parameters for gateway (field bus) units 10
3.3 Defining input and output signals 12
3.4 Defining signal groups 14
3.5 Defining cross connections 15
3.6 List all available I/O Unit Types 19
3.7 I/O Data Specifications 21
3.8 Defining system inputs 22
3.9 Defining system outputs 24
3.10 PLC Communication 26
4 Topic: Communication 31
4.1 Defining physical channels 31
4.2 Defining Transmission Protocol 32
4.3 Defining Application Protocol 34
5 Topic: Controller 37
5.1 Activate delayed safeguarded space stop 37
5.2 Activate Hold-To-Run Control 37
5.3 Defining event routines 38
5.4 Specifying regain distances 39
5.5 System miscellaneous 40
5.6 Automatic loading of modules and programs 41
5.7 Defining multitasking 42
6 Topic: TeachPendant 45
6.1 Defining Optional Packages 45

Page

6.2 Defining File Extension 45
6.3 Defining authorisation and confirmation 46
6.4 Activation of Limited ModPos Function 49
6.5 Programmable keys 50
6.6 Defining Running Mode Settings 51
7 Topic: Manipulator 53
7.1 Defining the commutation offset and calibration offset of the motors 53
7.2 Defining the range of movement and calibration position of each axis 54
7.3 Defining supervision level 54
7.4 Tuning the motion supervision 55
7.5 Defining teach mode speed 56
7.6 Defining independent motion 56
7.7 Defining arm load 57
7.8 Defining arm check point 58
7.9 Defining external torque 59
7.10 Defining friction compensation 60
7.11 Defining the base coordinate system 61
7.12 Defining external manipulators with more than one axis 62
7.13 Defining a track motion with coordinated motion 63
7.14 Defining an external mechanical unit coordinated with the robot 63
7.15 Defining external axes 63
7.16 Activate forced gain control for an external axis 75
7.17 Activate notch filter for an external axis 76
7.18 Soft servo for external axis 77
7.19 Defining the joystick directions for the robot and external manipulator 78
7.20 Defining the joystick directions for a single external axis 80
7.21 Defining kinematic parameters for general kinematics 81
7.22 Servo parameters 83
7.23 CPU Optimization 84
7.24 Installation optimization of drive system parameters 86

System Parameters

The *system parameters* describe the equipment and area of application of the robot system, e.g. I/O names and the characteristics of the external axes.

1 Changing a Parameter

1.1 Subdivision of parameters

The available parameters are grouped together in a number of different *topics*. These topics are, in turn, divided up into different *type*s.

Topic	Parameters that affect	File name
Controller	Event routines, etc.	SYS.CFG
Communication	Serial channels	SIO.CFG.
IO Signals	I/O boards and signals	EIO.CFG
Manipulator	The robot and external axes	MOC.CFG
TeachPendant	Displaying data and access on the teach pendant	MMC.CFG
Arc Welding	Arc welding	PROC.CFG

• To view all parameters, choose **Topics: All Topics** in the System Parameters window.

All relevant topics in the robot system will then be displayed. Choose the desired topic by selecting it and pressing Enter [↵] .

1.2 Changing a parameter

• Press the Miscellaneous key to open the System Parameters window.

• Select ***System Parameters*** from the dialog box that appears.

• Press ***OK*** or Enter [↵] .

• Call up the parameter type that contains the parameter to be changed, by choosing a topic from the **Topics** menu and a type from the **Types** menu.

All parameters of that type will be displayed, as illustrated in Figure 1. To be able to display some parameter types, however, you must first choose the current unit, such as an I/O board or a signal, by selecting it and pressing Enter [↵] .

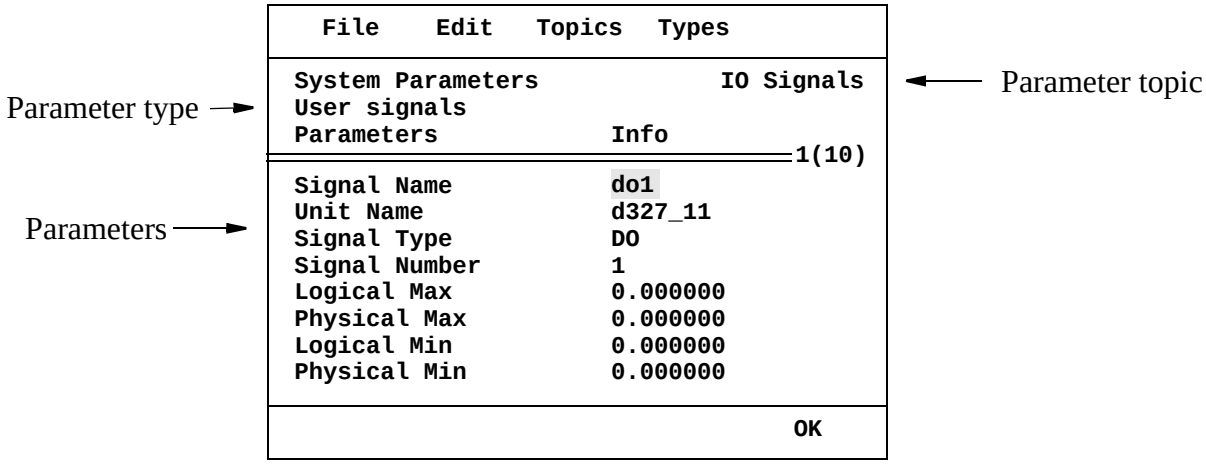


Figure 1 All parameters of a given type are displayed in the window at the same time.

- Select the parameter to be changed.
- Change the value of the parameter by
  - pressing Enter ↵ and specifying the desired alternative in the dialog box that appears,
  - choosing an alternative from the function keys (fields marked with ▫).

All parameters, together with possible values, are described in the following sections under the appropriate topic.

Note. You will have to restart the robot in order for the change to have an effect on some parameters. You will be informed of this the first time you change such a parameter and when you exit the system parameters, i.e. change window.

1.3 Deleting a parameter

- Select the parameter to delete
- Press ***Delete*** ⌫ .
- Press ***OK*** to confirm the delete.

1.4 Generating a restart

You have to restart the robot in order for a change to have an effect on some of the parameters. If you exit the system parameters without generating a restart, the parameter values will not be the same as those used in the robot. Nevertheless, if you generate a restart at a later stage, then the changes will take effect.

- Choose **File: Restart** and press ***OK*** or turn the mains switch off and then on again.

An error message will be displayed when there is an error in the parameters. However, this can be due to a sequential error. The origin of an error can be found by looking at the robot's error logs. See chapter 14, Service: *Logs*.

1.5 Viewing the last changes that were made

• Choose **Edit: Show Change Log**.

A dialog box appears, displaying the changes that were last made (see Figure 2).

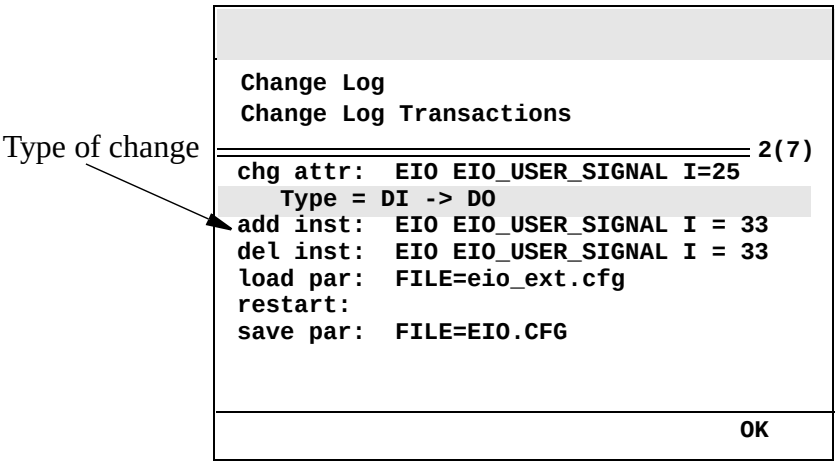


Figure 2 The Change Log dialog box.

The following identification tags are used:

- chg attr: Shows the parameter that has been changed and how it was changed.
- add inst: Shows that a new parameter has been added.
- del inst: Shows that a parameter has been deleted.
- load par: Shows that new parameters have been loaded.
- save par: Shows that parameters have been saved.
- restart: Shows that the robot has been restarted.

1.6 Checking Parameters

When you have changed a parameter, it is a sensible idea to check the change before restarting, in order to avoid problems when restarting. In the current version the Manipulator area can be checked.

• Select the area to be checked in **Topics** (only Manipulator can currently be checked)

• Select **File: Check Parameters** and the check will start.

When the check is finished, a report will be made showing that either there was an error or the change of the parameter was done correctly. The error will be reported via the usual error log. See chapter 14, Service.

2 Saving and Loading Parameters

2.1 Saving parameters to diskette or some other mass storage device

The system parameters can be stored in their entirety or stored as individual parameter topics, for example, on a diskette.

To save all parameters

The parameters are always saved to a directory in the form of a separate file for each topic.

• Choose **File: Save All As**.

A dialog box appears, displaying all files in the current directory (see Figure 3).

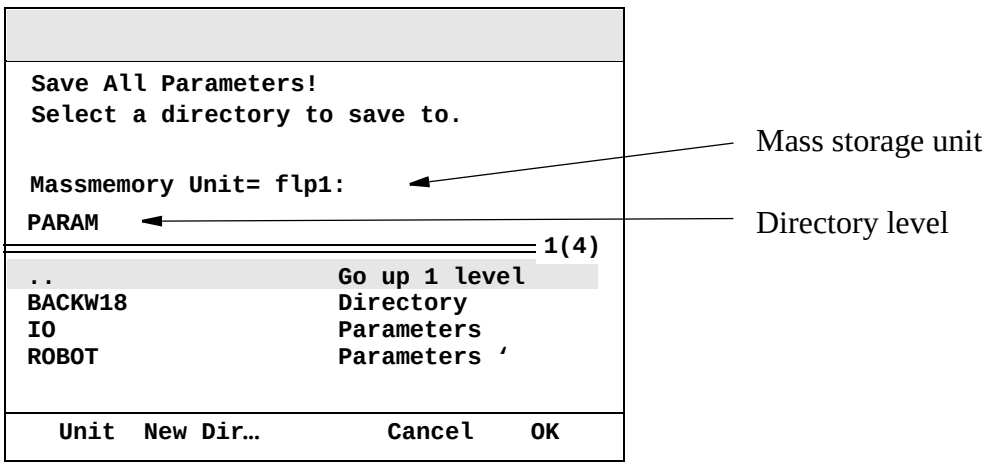


Figure 3 The dialog box used to store parameters.

• If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To store on a diskette, choose ***flp1:***.

• Select the directory to which the parameters are to be saved. You can move to the next directory level by selecting the desired directory or '. .' (upwards) and pressing Enter ↵.

Create a new directory by pressing the ***New Dir*** function key. Specify the new directory name in the dialog box that appears. Choose ***OK*** to confirm.

• Choose ***OK*** to confirm the save.

To save an individual parameter topic

• Choose **File: Save As**.

A dialog box appears, displaying all the previously saved parameters in the current directory.

- If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To store on a diskette, choose ***flp1:***.
- Specify the file name by selecting the field ***Name*** and pressing Enter ↵. Enter the desired name and press ***OK*** to confirm.
- Select the directory to which the parameters are to be saved. You can move to the next directory level by selecting the desired directory or '. .' (upwards) and pressing Enter ↵.
- Choose ***OK*** to confirm the save.

2.2 Loading parameters from a diskette or some other mass storage device

Parameters can be loaded in their entirety or loaded as individual parameter topics. If several parameters are to be loaded, the parameters must be placed in a directory.

- Choose **File: Load Saved Parameters** to replace a complete parameter topic.
- Choose **File: Add New Parameters** to add new parameters to the current one (e.g. when defining an external axis).
- Choose **File: Add or Replace Parameters** to replace parts of a parameter topic (e.g. when reediting an external axis). If a parameter already exists its value will be updated according to the new one. If the parameter does not exist it will be added.

A dialog box appears, displaying all parameters in the current directory (see Figure 4).

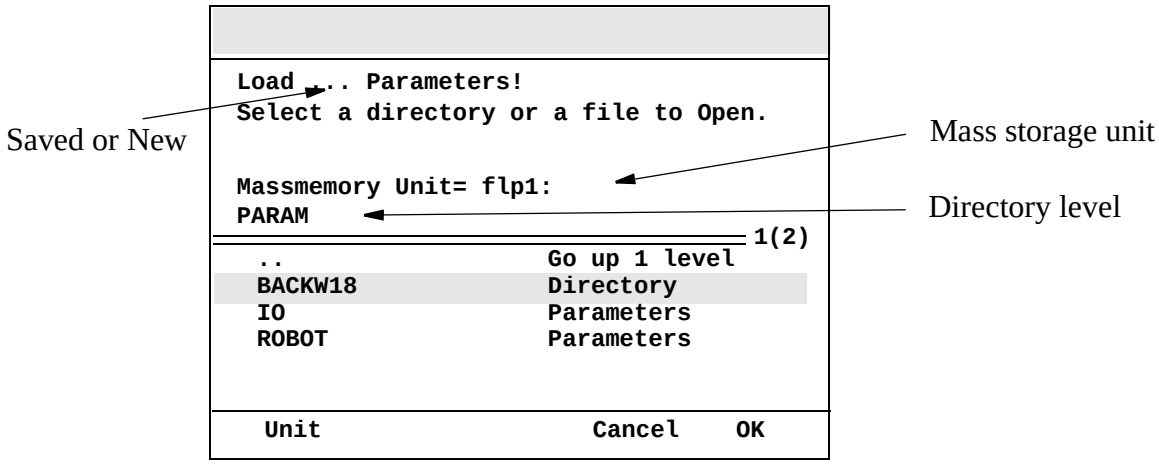


Figure 4 The dialog box used to load parameters.

- If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To load parameters from a diskette, choose ***flp1:***.
- Select the directory from which the parameters are to be loaded. You can move to the next directory level by selecting the desired directory or '. .' (upwards) and pressing Enter ↵.
- Choose ***OK*** to confirm the load.

An alert box will be displayed during reading. After this the robot must be restarted (see *Generating a restart* on page 4).

3 Topic: IO Signals

The following parameters are found under the topic IO Signals:

- Specification of all I/O units
- Name and characteristics of input and output signals
- Groups of digital signals
- Cross connections
- List of all available unit types
- System signals

• Choose **Topics: IO Signals**.

3.1 Defining I/O Units

• Choose **Topics: IO Signals**

• Choose **Types: IO Units**.

All defined units will be displayed, as shown in Figure 5.

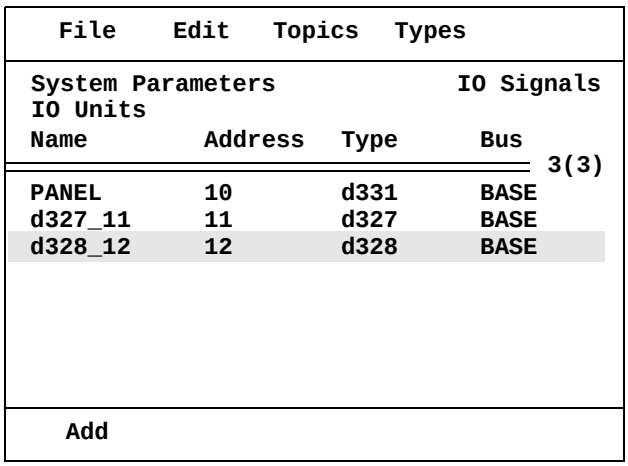


Figure 5 System parameters of the type IO Units.

• Select the appropriate I/O unit to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

To delete a unit

• Select the appropriate unit.

• Press ⌫.

All the signals on this unit will remain defined. These must be deleted separately.

Parameter	Description
Unit Name	The name of the unit (max. 16 characters).
Unit Type	Unit type (see also 3.2 below). Compare with the designation on the cover of the unit. Unit type *eip000* is used for simulated digital and analog signals. This simulated unit can handle up to 100 digital inputs, 100 digital outputs, 30 analog inputs and 30 analog outputs. For more information about the selected unit type press the function key ***Info***.
Unit Bus	The bus on which the unit is located, normally BASE. All units that are connected to the SIM bus will be simulated.
Address	The address of the unit on the BASE bus according to the physical keying, see Product Manual - *Installation and Commissioning.*
Digital Outputs	The number of digital output signals to be used.
Digital Inputs	The number of digital input signals to be used.
Analog Inputs	The number of analog input signals to be used.
Analog Outputs	The number of analog output signals to be used.
PollRate	Some units needs to be polled to update input and output signals. The parameter specifies the time between two consecutive polls. **NOTE**. All units produced by ABB Robotics Products, except the gateway (field bus) units, are event driven. With these units the polling mechanism is used as heart-beat only. It is recommended to use the default poll rate, 1 second, with these units (20 ms to10 s, resolution 10 ms).
Disabled	Specifies that the I/O unit will not be present on the bus at start up.

3.2 Additional parameters for gateway (field bus) units

A gateway unit is a unit which is connected to the internal BASE bus (Can/DeviceNet) and also to an external field bus, such as Interbus-S, Profibus, A-B RIO.
The gateway unit converts the data from one field bus to another field bus.

Allen-Bradley Remote I/O adapter DSQC 350 (Unit Type = d350)

Parameter	Description
Rack Address	The rack address is entered in decimal form while Allen-Bradley use octal (0-7). Valid values are from 0 to 63 (Allen-Bradley 0-77). Note that the Allen-Bradley PLC series 2/30 starts from octal address 1.
Data Rate	The data rate on the RIO Bus. Valid values are: 57.6 KBaud. 115.2 KBaud. 230.4 KBaud.

Starting Quarter	The DSQC 350 starting quarter.		
	Valid values are:	First	PLC value 0
		Second	2
		Third	4
		Fourth	6
Rack Size	The DSQC 350 rack size.		
	Valid values are:	1/4 rack	32 out, 32+2 in[1]
		1/2 rack	64 out, 64+2 in[1]
		3/4 rack	96 out, 96+2 in[1]
		full rack	128 out, 128+2 in[1]
Last Rack	Specifies that the DSQC 350 is the last rack on the RIO Bus.		
	Valid values are:	No	
		Yes	

Interbus-S slave adapter DSQC 351 (Unit Type = d351)

Parameter	Description
Data Width	The data width determines the number of I/O signals. The valid values are: 1 word = 16 out, 16+1 in[2] 2 words = 32 out, 32+1 in[2] 3 words = 48 out, 48+1 in[2] 4 words = 64 out, 64+1 in[2]

Profibus-DP slave DSQC 352 (Unit Type = d352)

Parameter	Description
Master Input Size	The master input size determines the number of digital output signals. The valid values are: 0 word = 0 out 1 word = 16 out 2 words = 32 out 3 words = 48 out 4 words = 64 out 5 words = 80 out 6 words = 96 out 7 words = 112 out 8 words = 128 out
Master Output Size	The master output size determines the number of digital input signals. The valid values are: 0 word = 0+1 in 1 word = 16+1 in 2 words = 32+1 in 3 words = 48+1 in 4 words = 64+1 in

1. Input numbers 33, 65, 97 or 129 respectively, correspond to NAC status LED flashing (high at RIO link particularly up, i.e. PLC in programming mode)
Input numbers 34, 66, 98 or 130 respectively, correspond to NAC status LED steadily lit (high at RIO link up)
2. Input numbers 17, 33, 49 or 65 respectively, correspond to Interbus-S status LED BA steadily lit (Bus Active)

5 words = 80+1 in
6 words = 96+1 in
7 words = 112+1 in
8 words = 128+1 in

Station Address The Profibus node address entered in decimal form. Valid values are from 2 to 126. 126 is reserved for uninitialised nodes.

To configure the Profibus master use the GSD-file supplied on the diskette *Controller Parameters*, file SERVICE\GSD\ABB_0600.GSD. The modules to be choosen depends on the I/O definition.

3.3 Defining input and output signals

• Choose **Topics: IO Signals**.

• Choose **Types: User Signals**.

All named signals will be displayed (see Figure 6).

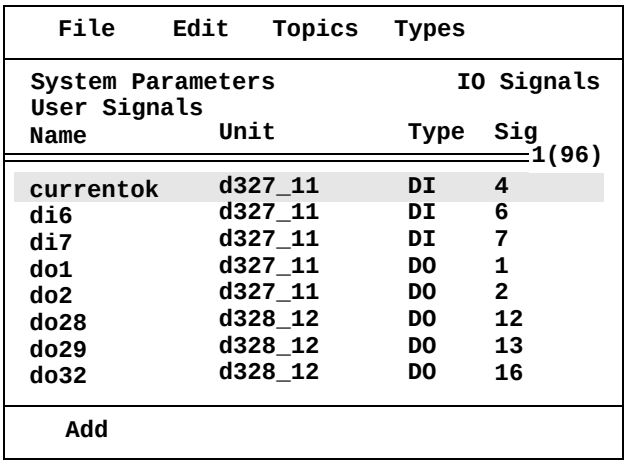


Figure 6 System parameters of the type User Signals.

Note that several signals can be connected to the same physical channel. The same signals cannot, however, be connected to different physical channels.

• Select the signal to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

To delete a signal

• Select the appropriate signal.

• Press ⌫.

Parameter	Description
Signal Name	The name of the signal (max. 16 characters).
Unit Name	The unit to which the signal is connected. For more information about the selected unit press the function key ***Info***.
Signal Type	The type of signal: ***DI***= Digital Input, ***DO***= Digital Output, ***AI***= Analog Input, ***AO***= Analog Output.
Signal Number	The physical channel to which the signal is connected. The numbering restarts from 1 for each unit and "signal type".
Logical Max ***Physical Max*** ***Logical Min*** ***Physical Min***	The scaling between the programmed and physical value of an analog signal (see Figure 7).

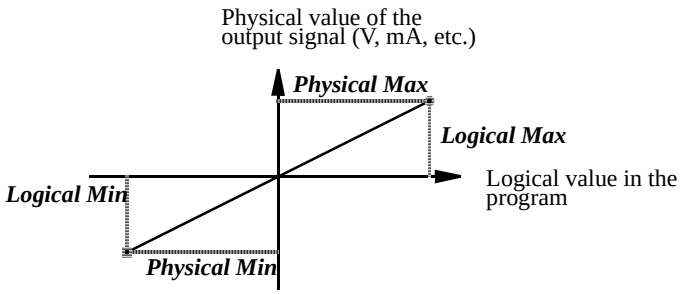


Figure 7 Diagram to show how analog signal values are scaled

Logical max/min. is the maximum/minimum value that an analog input or output can be set to, from a RAPID program or from the teach pendant. The units are user defined (e.g. meter/minute).

Physical max/min. is the maximum/minimum physical value that can be set on the output or input. To obtain the physical limit for a specified unit, see the Product Manual.

If both physical and logical max/min. are set to 0, the default values for the unit are picked up, which are the physical maximum and minimum limits. The logical and physical is set to the same value, which gives an amplification factor of 1.
If any of the values is set by the user, all four must be defined. Therefore, make sure that:

physical maximum is > physical minimum
logical maximum is > logical minimum.

Example: An analog unit is controlling a current source with an amplification of 50 A/V and a max current of 500A. The following settings could then be applicable.

Physical Max = 10 V
Physical Min = 0 V
Logical Max = 500 A/V
Logical Min = 0 A/V

Filter Passive	The time (in millisecs) that a digital input signal must be zero, before the robot acts on the signal change (100 ms to 32 s).
Filter Active	The time (in millisecs) that a digital input signal must be 1, before the robot acts on the signal change (100 ms to 32 s). If two signals are connected to the same physical channel, the filter time for these signals must be the same.
Inverted	Set to YES, if the digital signal shall be inverted, i.e. if logical ”1” should be set on the output as ”0”.
Store	If set to YES, the digital outputs will be stored at a power failure, and restored when the system is powered up again. It should be noted that the value is connected to a logical signal. If more than one logical signal is connected to the same physical signal, an unexpected value may be obtained. In such cases this parameter should be set to NO.

Maximum number of user defined signals including group signals = 512.
Signals connected to a simulated unit (Type = *eip000*) can not use Delay, Pulse or any time related instruction, only “clean” set and reset of digital input signals.

3.4 Defining signal groups

Digital signals can be grouped together and handled as if they were one signal. The value of such a signal will thus be a positive integer that is binary coded using the individual digital signals as a basis.

- Choose **Topics: IO Signals**.
- Choose **Types: Groups**.

All defined signal groups will be displayed (see Figure 8).

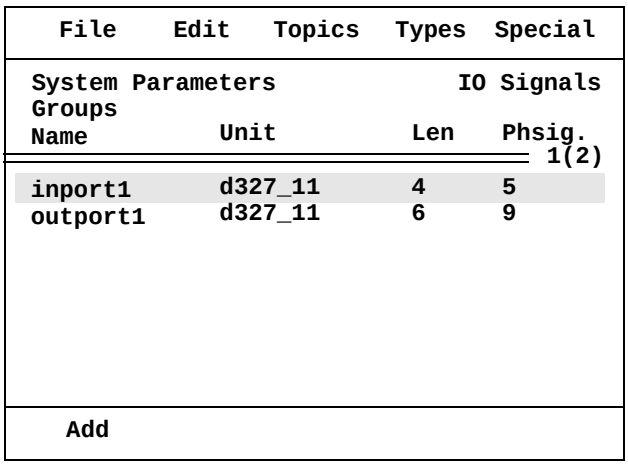


Figure 8 System parameters of the type Groups.

- Select the signal group to be changed, and press Enter ↵, or add a new one by pressing ***Add***.
- Select the desired parameter and change its value.

• Press ***OK*** to confirm.

To delete a signal group

• Select the appropriate signal group.

• Press ⌫.

Parameter	Description
Signal Name	The name of the signal (max. 16 characters).
Unit Name	The unit to which the signal is connected. For more information about the unit type press the function key ***Info***.
Signal Type	The type of signal: ***GI***= Group of input signals, ***GO***= Group of output signals.
Group length	The number of digital signals in the group. The length must be set so that the group is within one unit. The maximum value for length is 16.
Start signal	The physical channel of the unit to which the first signal (the least significant) is connected. The remaining signals are automatically connected to the following channels on the same unit.
Inverted	Set to Yes if all signals in the group shall be inverted.
Store	If set to YES, the digital outputs will be stored at a power failure, and restored when the system is powered up again. It should be noted that the value is connected to a logical signal. If more than one logical signal is connected to the same physical signal, an unexpected value may be obtained. In such cases this parameter should be set to NO.

3.5 Defining cross connections

A digital input or output signal can be logically connected to one or several digital input or output signals. This means that a cross-connected signal will automatically be changed when the “activation” signal is changed. See also RAPID Reference Manual - *Motion and I/O Principles*.

• If the signal has not already been defined, then define its name in the normal way. See *Defining input and output signals* on page 12.

• Choose **Topics: IO Signals**

• Choose **Types: Cross Connections**.

All the defined cross connections will be displayed (see Figure 9).

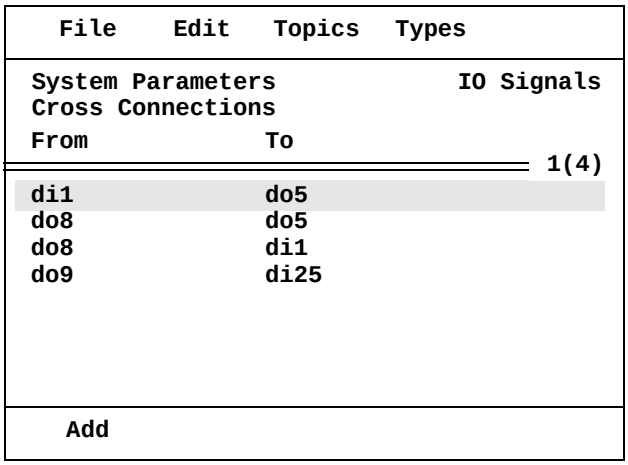


Figure 9 An output signal can be logically connected to an input signal.

- Select the cross connection to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Define the ”activation (From)” signal and the corresponding ”cross connected (To)” signal.
- Press ***OK*** to confirm.

A maximum of 60 signals can be cross connected. Make sure that the same signal is not connected on both the ”From” and ”To” sides, in the same chain.

To delete a cross connection

- Select the appropriate cross connection.
- Press ⌫.

Cross connections with logical conditions (Option)

The digital I/O signals can have the logical conditions AND or OR, to set up a condition for a cross connection. These conditions cannot be entered from the teach pendant. They are instead set up in the configuration file EIO.CFG in the cross-connection section (starting with the line “EIO_CROSS:”) using a standard PC. The same rules apply to the logical condition connections for the result signals as for the normal cross-connected result signals. The actors in the cross-connection section have the logical condition operators.

The logical condition operators are:

- AND, syntax in configuration file = “&”
- OR, syntax in configuration file = “!”

For each logical condition connection there can only be one kind of logical operator. Each logical condition connection can be seen as a logical operator box.

The AND operator has the following function:

- If **all** in signals (actor signals) to the AND box are high, the result signals will be high.

The OR operator has the following function:

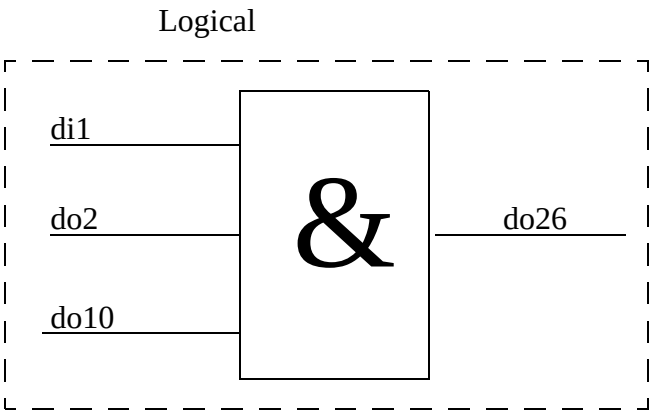
- If **any** in signals (actor signals) to the OR box are high, the result signals will be high.

There is one help operator:

- INV, syntax in configuration file = “*” , inverted.

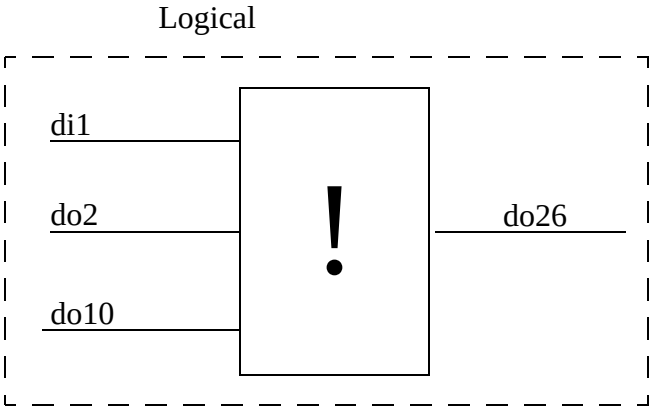
The INV help operator can be connected before an in signal (actor signal) to an AND or OR box which means that the signal is inverted before being checked in the operator box.

Examples



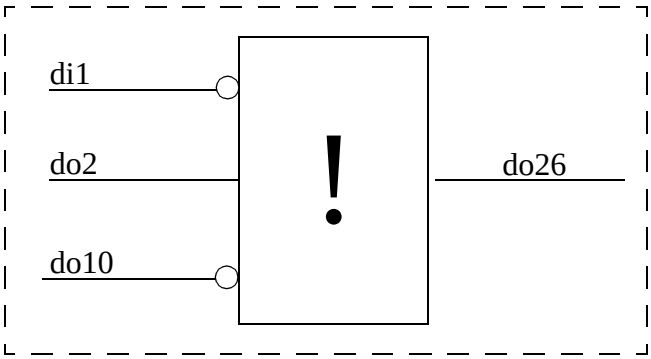
In Configuration file:

EIO_CROSS:
-Lres do26 -Lact di1 & do2 & do10



In Configuration file:

EIO_CROSS:
-Lres do26 -Lact di1 ! do2 ! do10

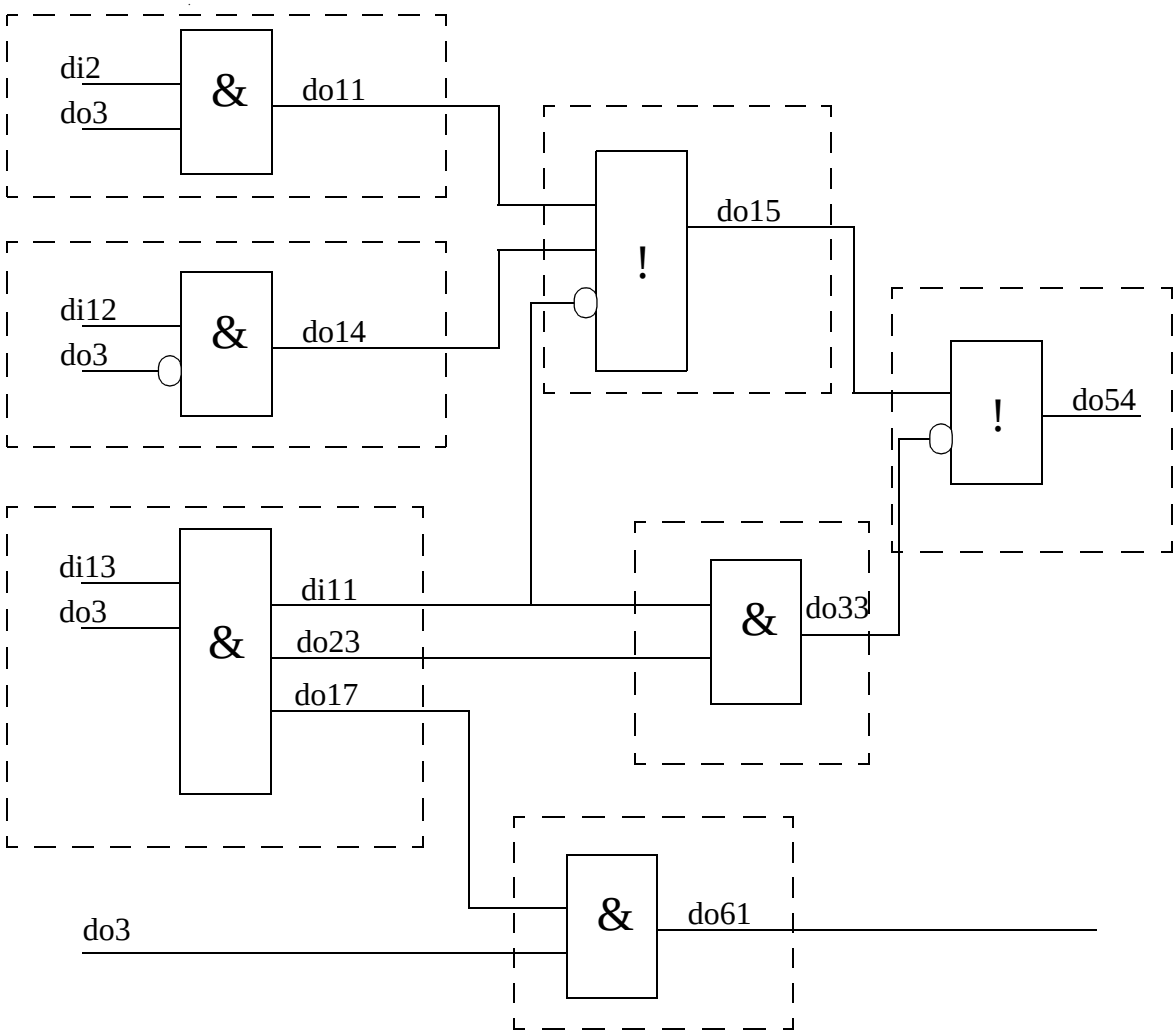


○ = INV = Invert

In Configuration file:

EIO_CROSS:
-Lres do26 -Lact *di1 ! do2 ! *do10

The actor signals can be both digital In and Out signals. There can be 5 actor signals in each condition but there can be several conditions. The cross-connected signals cannot use Delay or Pulse or any parameters, only “clean” set and reset of digital in and out signals. The example below describes a configuration file that has several logical condition connections



In Configuration file:

```
EIO_CROSS:
-Lres do11      -Lact di2 & do3
-Lres do14      -Lact di12 & *do3
-Lres di11 + do23 + do17      -Lact di13 & do3
-Lres do15      -Lact do11 ! do14 ! *di11
-Lres do33      -Lact di11 & do23
-Lres do61      -Lact do17 & do3
-Lres do54      -Lact do15 ! *do33
```

Calculation of number of cross connections

If you make a cross connection with logical conditions, the number of cross connections may be more than the number of signals actual used in the cross connection itself.

You will use one cross connection for each input to a gate, and one for the output of the gate. If you connect an output from one gate to an input of another gate, it takes two cross connections even if there is only one signal involved.

In the configuration above you have 15 signal names used, but it takes 24 cross connections.

3.6 List all available I/O Unit Types

• Choose **Topics: IO Signals**

• Choose **Types: Unit Types**

All available unit types will be displayed, as shown in Figure 10.

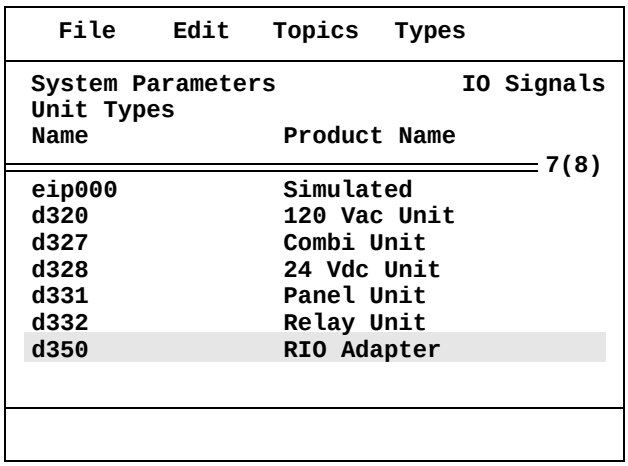


Figure 10 List of all available unit types.

To get detailed information about a particular unit type:

• Select the appropriate unit type and press Enter ↵

Parameter	Description

Unit Type	Name of the unit type
Vendor	Name of the vendor that manufactures this unit type
Product	Description of this unit type
Vendor ID	Vendor's identification code
Product ID	Unit's product identification
Device Type ID	Unit type identification code
Major Revision	Major revision for the software resident in the unit
Minor Revision	Minor revision for the software resident in the unit
Digital Inputs	Maximum number of digital input signals available on this unit type
Digital Outputs	Maximum number of digital output signals available on this unit type
Analog Inputs	Maximum number of analog input signals available on this unit type
Analog Outputs	Maximum number of analog output signals available on this unit type

Adding User Defined I/O Unit Types (Option)

I/O devices that does not have a unit type equivalent listed on the teach pendent, needs to be defined in the EIO.CFG configuration file. This cannot be done from the teach pendant. It has to be done using a standard PC.

User defined unit types are defined in the unit type section of the configuration file. Create this section by writing "EIO_UNIT_TYPE:" at the end of the file.

The unit types are defined in the configuration file, using the syntax below filling in the appropriate *italic*- value (the figures below are only examples):

EIO_UNIT_TYPE:
-Name *typename* \
-VendorId *75* \
-VendorName *"Vendor Name"* \
-DeviceType *7* \
-ProductId *334* \
-ProductName *"Product Name"* \
-MajorRev *2* \
-MaxDigin *16* \
-MaxDigout *16* \
-MaxAnin *0* \
-MaxAnout *0*

Some devices may respond incorrectly to commands sent from the robot controller during connection establishment. The robot controller will then generate the error message: 71102 "DeviceNet incompatible".

This can be avoided by explicitly specifying that the controller should use a more trivial method for connection establishment. This is done by adding the line; '*-DeviceDriver DNGeneric*' to the configuration file:

EIO_UNIT_TYPE:
-Name *typename* \
-VendorId *75* \
-VendorName *"Vendor Name"* \
-DeviceType *7* \
-DeviceDriver "DNGeneric"\
-ProductId *335* \
-ProductName *"Product Name"* \
-MajorRev *2* \
-MaxDigin *16* \
-MaxDigout *16* \
-MaxAnin *0* \
-MaxAnout *0*

3.7 I/O Data Specifications

The names of I/O signals, units etc. have a maximum limit of 16 characters.

Each name must be unique in the topic IO Signals.

I/O signals connected to a simulated I/O unit cannot use Pulse, Delay or Filter.

A maximum of 20 I/O units, including simulated I/O units can be defined.

A maximum of 10 simulated I/O units can be defined.

“Pollrate” for I/O units has the range:

- Minimum: 10 ms
- Maximum 10 seconds
- Step: 10 ms

A maximum of 512 signal names, including user signals and group signals can be defined.

Filter Passive and Active time has the range:

- Minimum: 100 ms
- Maximum: 32 seconds
- Step: 10 ms

A maximum of 128 I/O signals can be defined with the “Store” parameter.

A maximum of 60 cross connections can be defined.

3.8 Defining system inputs

The input signals can be assigned specific system actions. In this case, they will automatically be handled by the robot. See also *PLC Communication* on page 26. System inputs with actions including a MOTORS ON order or Start of any robot movement are allowed in automatic mode only. When a system input order is rejected because the system is in manual mode or due to any other unfulfilled requirement, no error messages will be displayed on the teach pendant.

The reason for this is that the teach pendant is normally not being used by the operator in these situations and there is therefore no sense in displaying the messages on the teach pendant. However, the messages will still be stored in the error log, so you may check for an answer there as to why a system input gives no action.

- If the signal has not already been defined, define its name in the normal way. See *Defining input and output signals* on page 12.
- Choose **Types: System Inputs**.

All defined system inputs will be displayed (see Figure 11).

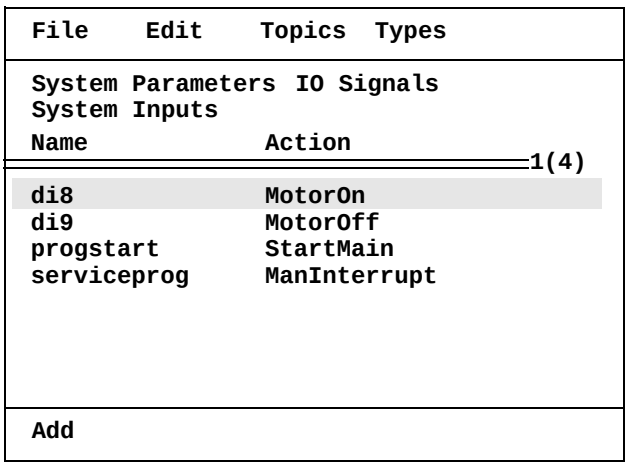


Figure 11 Input signals can be assigned specific system actions.

- Select the system input to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Define the name of the signal and the system action that is assigned to it. To add the system actions *MotOnStart, Start, StartMain, Interrupt, LoadStart* and *Sync ExtAx*, define their arguments as well.
- Press ***OK*** to confirm.

The following system actions are available:

System action	Description
MotorOn	Sets the robot to the MOTORS ON state.

MotOnStart	Sets the robot to the MOTORS ON state and starts the program (continuous or cycle execution) from the current instruction, i.e. from the program pointer.
MotorOff[1]	Sets the robot to the MOTORS OFF state. If a program is executing, it will be stopped before changing state. The robot cannot be set to the MOTORS ON state when this signal is high.
Start	Starts the program (continuous or cycle execution) from the current instruction, i.e. from the program pointer.
StartMain	Starts the program (continuous or cycle execution) from the beginning. Not valid during program execution.
Stop[1]	Stops program execution. A program cannot be started when this signal is high.
QuickStop[1]	Stops program execution. Gives a quicker stop and some deviation from the programmed path. A program cannot be started when this signal is high.
StiffStop[1]	Stops program execution. Gives an even quicker stop and some more deviation from the programmed path than QuickStop. A program cannot be started when this signal is high.
StopInstr[1]	Stops program execution after the current instruction has been finished. A program cannot be started when this signal is high.
StopCycle[1]	Stops program execution when the complete program has been executed, i.e. when the last instruction in the main routine has been executed. A program cannot be started when this signal is high.
SysReset	Performs a system reset (restart), i.e. similar to power off/on.
Interrupt	Executes a routine (procedure) without changing the start pointer. Not valid during program execution. When the routine has been executed, the normal program can be restarted. The name of the routine to be executed is also defined in this dialog, e.g. routine1. This signal, for example, can be used to go to a service position. When the normal program is started again, the robot will move to the point where it was stopped and continue from there.
ResetError	Resets the system output signal Error.
SyncExtAx	Synchronises an external mechanical unit. The unit to be synchronised is also defined in this dialog e.g. orbit1. One signal is required for each unit. (S4 only)
LoadStart	Loads a program from diskette or other mass storage device. The program is then started from the beginning. The program file name (including mass memory unit) to be

	loaded is also defined in this dialog, e.g. flp1: PROGRAM1.PRG. Not valid during program execution.
ResetEstop	Resets the emergency stop. The robot can then be set to the MOTORS ON state.
AckErrDialog	Acknowledge the error dialogue on the teach pendant, i.e. the same operation as pressing OK on the teach pendant.

Several input signals can be assigned the same system actions, but several system actions may not be assigned to the same signal.

1. Allowed in both manual and automatic mode.

To delete a system action

• Select the signal to be deleted.

• Press ⌫

The system action assigned to this signal is then deleted, but the signal itself remains defined.

3.9 Defining system outputs

The output signals can be assigned a specific system status. In this case, they will automatically be handled by the robot.

• If the signal has not already been defined, define its name in the normal way. See *Defining input and output signals* on page 12.

• Choose **Types: System Outputs**.

All defined system outputs will be displayed (see Figure 12).

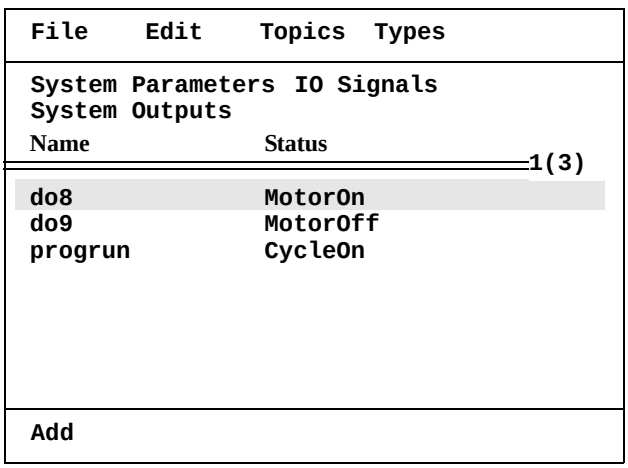


Figure 12 Output signals can be assigned specific types of system status

- Select the system output to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Define the name of the signal and the system action assigned to it.
- Press ***OK*** to confirm.

The following types of system status are available:

System status	Description
MotorOn	The robot is in the MOTORS ON state. If the robot system is not synchronised, the output will start flashing.
MotOnState	The robot is in the MOTORS ON state. The output is stable, i.e. no flashing.
MotorOff	The robot is in the MOTORS OFF state. If the safety chain is broken, the output will start flashing.
MotOffState	The robot is in the MOTORS OFF state. The output is stable, i.e. no flashing.
CycleOn[2]	A program is executing.
EmStop	The robot is in the Emergency Stop state. The emergency stop must be reset before the robot can be set to the MOTORS ON state.
AutoOn	The robot is in automatic mode.
RunchOk	The run chain is not broken.
TCPSpeed	An analog signal that describes the speed of the TCP. The logical value of the signal is specified in m/s, e.g. a speed of 2000 mm/s corresponds to the logical value 2. The scaling factor for the physical value is specified in the system parameters of the corresponding signal.
Error	The robot program execution has been stopped due to an error. (If an error occurs when a program is not executing, this output will not be set.)
PFError	A power failure error has occurred. The program would not restart after this type of error. The program can usually be started, but it will start from the beginning.
MotSupTrigg	The motion supervision function has been triggered.
MotSupOn	The motion supervision function is active.
RegDistErr	An attempt to start the program has been done but this failed due to that the robot was outside the Regain Distance zone (i.e. too far from the programmed path). The output is reset by one of the following actions: - A restart of the program has to succeed (the robot must be jogged into the zone first) - The PP has to be set to Main.

Several output signals can be assigned the same system status, but several system status may not be assigned to the same signal.

2. Also active for Event Routine execution.

To delete a system status

• Select the signal to be deleted.

• Press ⌫.

The system status assigned to this signal is then deleted but the signal itself remains defined.

3.10 PLC Communication

This chapter describes how to control the robot using, for example, digital signals from a PLC.

For a definition of the signals see chapter *Defining system inputs* on page 22 and *Defining system outputs* on page 24.

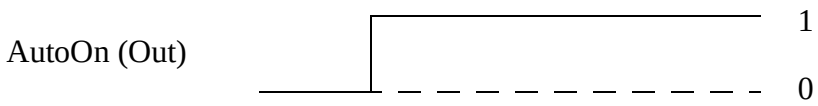
All system inputs are 0 to 1 level sensitive and the pulse length must exceed 50 ms.

Most system inputs are only permitted in the automatic mode. If any interactive unit, such as the teach pendant, a computer link, etc., has reserved exclusive rights to one or more functions in the system, the system input request will be denied. The signals MotorOff and Program stop signals from the system Input are active in Manual mode also.

Some examples of the signal sequences are described below.

To verify that the robot is in automatic mode

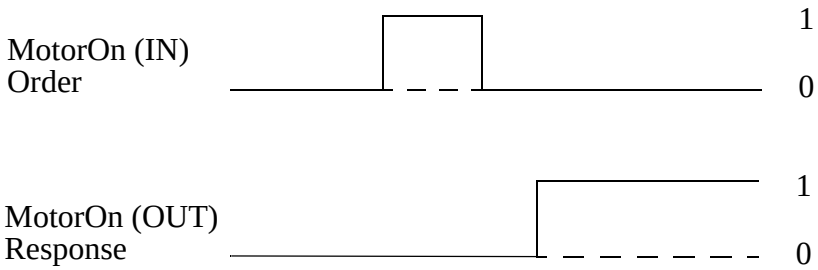
Signal sequence:



To switch the robot to MOTORS ON state

Requirement: Robot in MOTORS OFF state and Runchain is closed (=RunchOK)

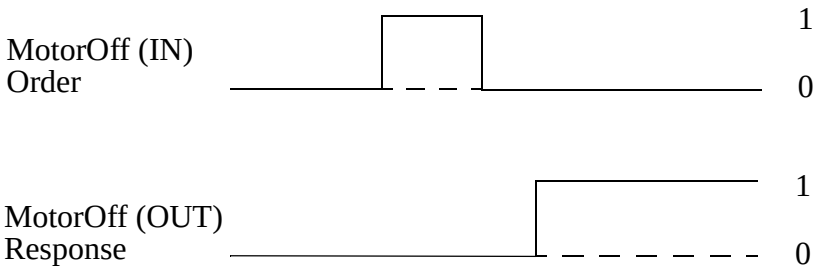
Signal sequence:



To switch the robot to MOTORS OFF state

Requirement: Robot in MOTORS ON state.

Signal sequence:

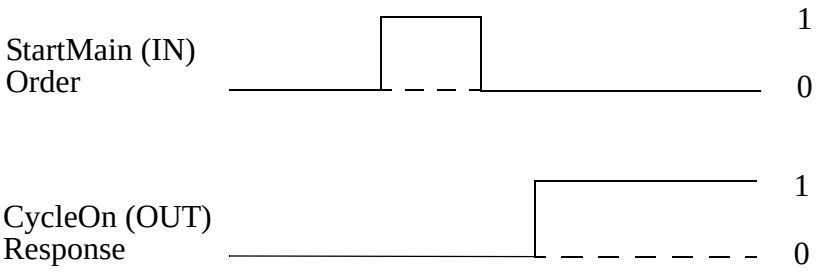


If the program is running (CycleOn), the MotorOff action will stop execution of the program.

To start the program from the beginning of the main routine

Requirement: Robot in MOTORS ON state and program control not occupied by any other resource (e.g. external computers).

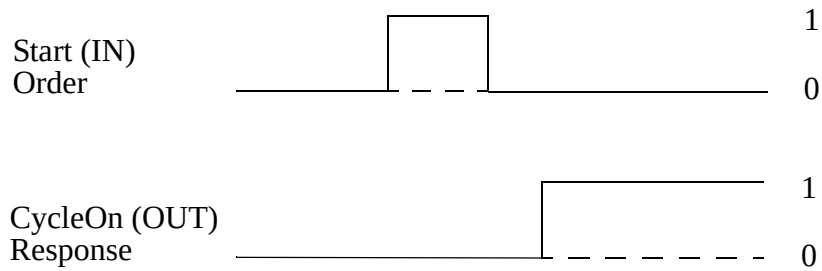
Signal sequence:



To start or restart program execution from the current instruction or after a power failure

Requirement: Robot in MOTORS ON state and program control not occupied by any other resource (e.g. external computers). A PFError indicates that a power failure error has occurred

Signal sequence:

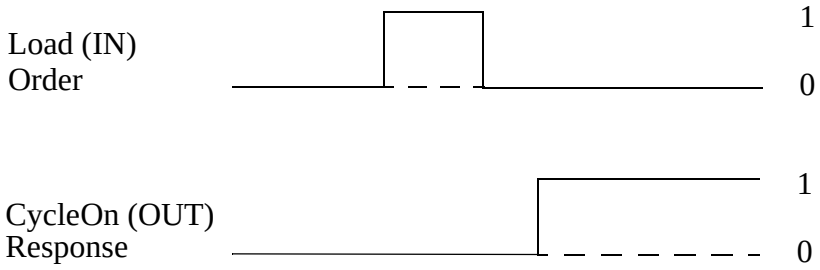


To load and start a program

Load a program from diskette or another mass storage device. The program will then start from the beginning. If a program is running, execution will stop first.

Requirement: Robot in MOTORS ON state and program control not occupied by any other resource (e.g., external computers).

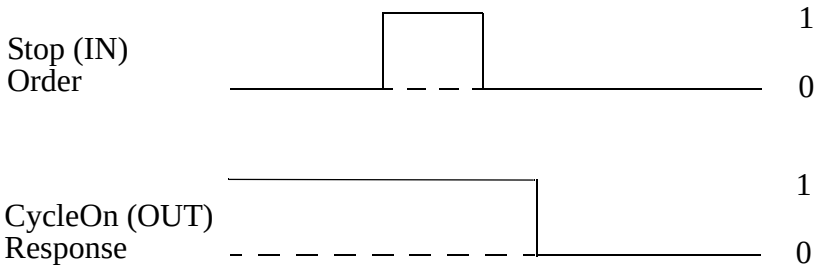
Signal sequence:



To stop, QuickStop, StiffStop program execution

Requirement: Valid in all modes.

Signal sequence:

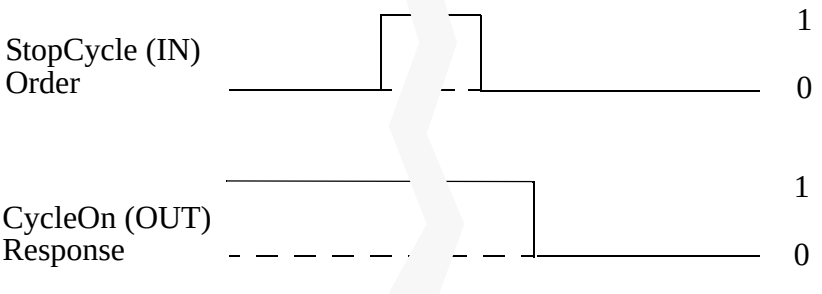


To stop at the end of the cycle

Stops program execution when the complete program cycle has been executed.

Requirement: Valid in all modes.

Signal sequence:



To detect spontaneous execution stops

Requirement: Robot in AutoOn (1), MotorOn (1) and CycleOn (1).

Signal sequence:

There are three main reasons why stops occur:

1. Program controlled exit, stop (or error in the program).

2. Emergency stop.

3. Safety chain broken due to reasons other than an emergency stop.

Detect case 1 with:
Error (1).

Detect case 2 with:
MotorOn (0), CycleOn (0), EmStop (1) and RunchOK (0).

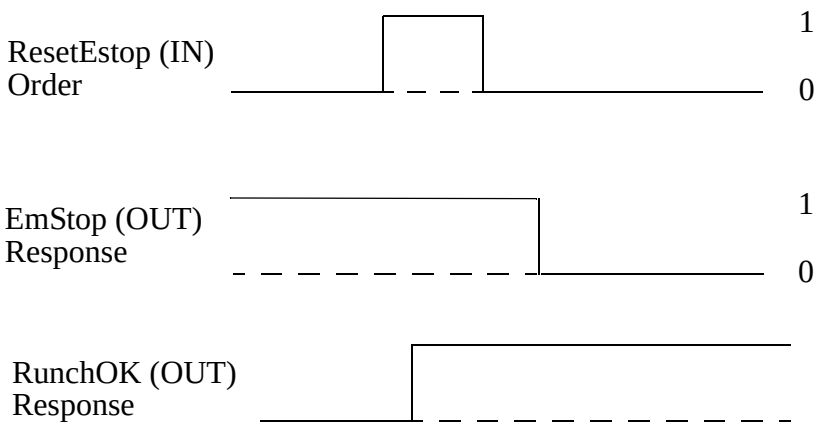
Detect case 3 with:
MotorOn (0), CycleOn (0), EmStop (0) and RunchOK (0).

To reset an emergency stop

Switches the robot back to MOTORS OFF state after a spontaneous emergency stop.

Requirement: Robot in automatic mode after an emergency stop. The safety chain must be closed, by resetting the emergency stop.

Signal sequence:



Continue by switching the power to the motors back on.

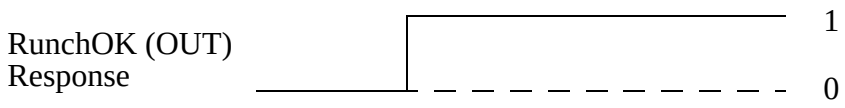
RunchOK

Indicates that the robot is back to MOTOR OFF state after a spontaneous safety chain stop.

Requirement: Robot in automatic mode and spontaneous stop case 3 (see above).

Signal sequence:

Wait until the RunchOK is high (the safety chain is closed).



Continue by switching the power to the motors back on.

4 Topic: Communication

The following parameters are found under the Communication topic:

- Specification of physical channels.
- Transmission protocols.
- Application protocols.

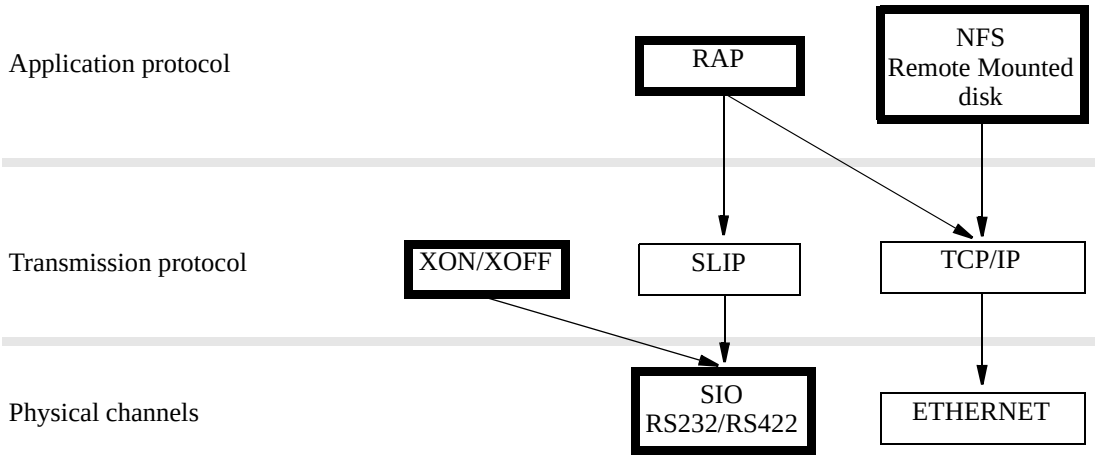


Figure 13 The following combinations of application protocols, transmission protocols and physical channels are possible. The boxes with a thick frame can be an endpoint of a configuration, i.e. they can function independently of the overlaying layers.

• Choose **Topics: Communication**

4.1 Defining physical channels

• Choose **Topics: Communication**

• Choose **Types: Physical Channels**

All defined physical channels will be displayed, as shown in Figure 14.

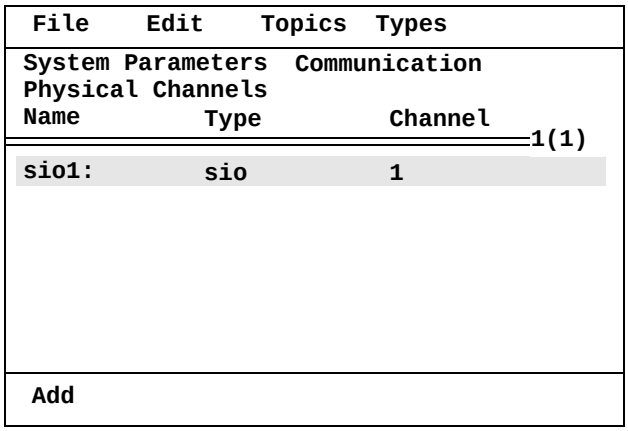


Figure 14 System parameters of the type Serial Channels.

- Select the physical channel to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Select the desired parameter and change its value. Press ***OK*** to confirm.

Parameters	Description
Name	Name of physical channel (max. 16 characters)
Type	Type of physical channel (SIO, ETHERNET). See Figure 13.

SIO - Serial Channels

Additionally, the following parameters must be set if the **Type** is set to *SIO*

Channel	Channel number (1 - 2).
Baud rate	Baud rate for the serial channel (300 - 19200).
Parity	Type of parity for serial channel. (Odd, Even, None).
No / of bits	Number of data bits (7, 8).
No / of stops bits	Number of stop bits (1, 2).
RTS / CTS Control	RTS / CTS flow control when sending from the robot (ON/OFF). RTS / CTS is not available on channel 2.

4.2 Defining Transmission Protocol

- Choose **Topics: Communication**
- Choose **Types: Transmission Protocol**

All defined transmission protocols will be displayed, as shown in Figure 15.

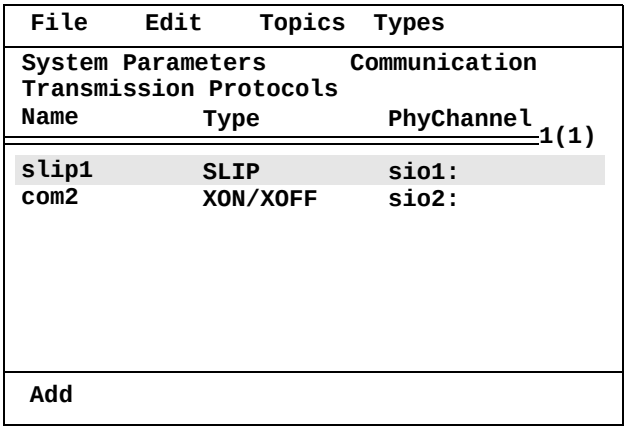


Figure 15 System parameters of the type Transmission protocol.

- Select the transmission protocol to be changed and press Enter ↵ , or add a new one by pressing ***Add***.
- Select the desired parameter and change its value. Press ***OK*** to confirm.

Parameters	Description
Name	Name of the transmission protocol (max. 16 characters). The name must be unique and must not be used anywhere else.
Type	Type of transmission protocol (None, XON/XOFF, SLIP, TCP/IP). See Figure 13.
PhyChannel	Name of the physical channel the protocol should use.

SLIP

Additionally, the following parameters must be set if the **Type** is set to *SLIP*

Local Address	Local address of the SLIP connection.
Remote Address	Remote address of the SLIP connection.

TCP/IP

Additionally, the following parameters must be set if the **Type** is set to *TCP/IP*

Local Address	Local IP address. Note that the robot will use 4 IP addresses. This address, ta+1, +2, +3 e.g. “192.168.1.1”, “192.168.1.2”, “192.168.1.3” and “192.168.1.4”.
Subnet Mask	The subnet mask of the ethernet interface.
XON/XOFF	This protocol can only be used if ASCII values are transmitted, i.e. to a terminal or a printer.

4.3 Defining Application Protocol

- Choose **Topics: Communication**
- Choose **Types: Application Protocol**

All defined application protocols will be displayed, as shown in Figure 16.

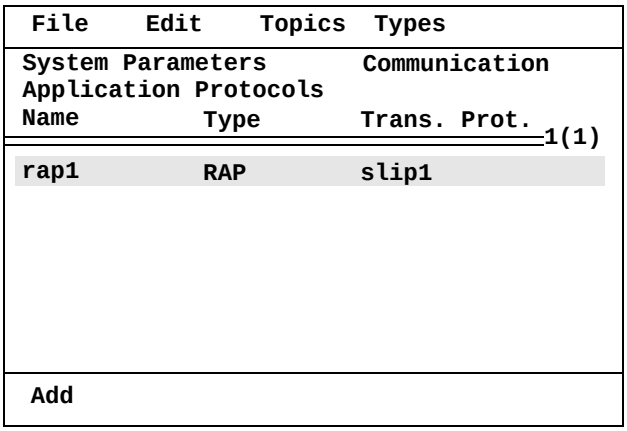


Figure 16 System parameters of the type Application protocol.

- Select the application protocol to be changed and press Enter ↵, or add a new one by pressing ***Add***.
- Select the desired parameter and change its value. Press ***OK*** to confirm.

Parameters	Description
Name	Name of the application protocol (max. 16 characters). The name must be unique and must not be used anywhere else.
Type	Type of application protocol (RAP, NFS). See Figure 13.
Trans Prot.	Name of the transmission protocol the protocol should use.

RAP - Robot Application Protocol

Additionally, the following parameters must be set if the **Type** is set to *RAP*

Remote Address	The IP Address of the Remote Computer. This variable is required when sending start up messages. If the transmission protocol is SLIP this parameter can be left empty. The *Remote Address* of the SLIP connection is used instead.
PortNo	The TCP protocol port number of the remote computer.
Send start-up msg	When enabled the robot controller can send a message at start up.
Enable SUBSCW	This variable is used for setting up subscriptions of RAPID events.

NFS - Remote Mounted Disk

Additionally, the following parameters must be set if the **Type** is set to *NFS*

Server Address	The IP Address of the computer with the NFS server.
Trusted	This flag decides if this computer should be trusted, i.e. if losing the connection should make the program stop. (YES/NO)
Local Path	The name of the unit on the robot. If, for example the unit is named *pc:*, the name of the *test.prg* on this unit would be *pc:test.prg*
Server Path	The name of the exported disk on the remote computer.
User ID	The UNIX user id for the mounting. (This parameter must be the same for all mountings on a robot)
Group ID	The UNIX group id for the mounting. (This parameter must be the same for all mountings on a robot)

Show on Teach Pendant

This flag decides if the unit should be visible in the list of units on the teach pendant. (YES/NO)

5 Topic: Controller

The following parameters are found under the Controller topic:

- Activation of delayed (soft) safeguarded space stop
- Activation of Hold-To-Run Control
- Event routines
- Maximum regain distances
- System miscellaneous
- Automatic loading of modules and programs
- Tasks (option Multitasking is required)

• Choose **Topics: Controller**.

5.1 Activate delayed safeguarded space stop

A delayed stop gives a smooth stop. The robot stops in the same way as a normal program stop with no deviation from the programmed path. After approx. 1 second the power supplied to the motors shuts off. The stopping distance can be longer than at a hard stop (e.g. emergency stop).

• Choose **Types: SafetyRunChain**.

• Select the safety function to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Define the function and whether or not it shall be active ***(True)***.
SoftAS = Delayed automatic mode safeguarded space stop
SoftGS = Delayed general mode safe guarded space stop

• Press ***OK*** to confirm.

5.2 Activate Hold-To-Run Control

When using the Hold-To-Run control, the program start key must be held down all the time, in order to execute a program.
This function is always activated in the manual operating mode at full speed, but can also be activated at reduced speed.

• Choose **Types: SafetyOpKey**.

• Select the safety function to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Define the function and whether or not it shall be active ***(True)***.

• Press ***OK*** to confirm.

5.3 Defining event routines

Special system events, such as program stop, can be connected together with an ordinary routine. When the event occurs, the corresponding routine is executed automatically.

• Choose **Types: Event Routines.**

All defined event routines will be displayed (see Figure 17).

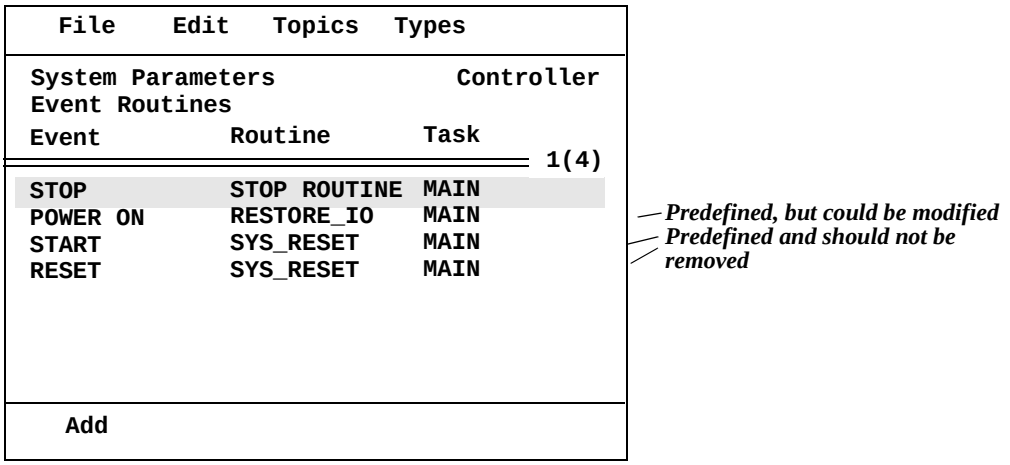


Figure 17 Certain events can start routines automatically.

• Select the event routine to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Define the system event and the routine assigned to it, also select which task the definition is for.

• Press ***OK*** to confirm.

The following types of system events are available:

System event	Description
POWER ON	The robot is restarted (warm start) from the teach pendant or by power on.
START	Execution is started from the beginning of the program.
RESTART	Execution is started from the position where is was stopped.
STOP	The program was stopped. **Note:** A delayed stop after current cycle will not execute the routines connected to this state.
QSTOP	The robot was quick stopped (emergency stop).
RESET	The old program was erased.

The specified routine must be a procedure without any parameters. The routines should be in a system module, at least the routines for the RESET event.

If the robot cannot find the specified routine, an error message will be given.

Avoid motion instructions in the routines. For STOP/QSTOP, a motion instruction in the corresponding event routine will result in an error. It is advisable to keep the routines short and quick.

A maximum of four routines may be specified for each system event and each task (multitasking). The same routine can be used in more than one event.

If there is a *Stop* or a *Break* instruction in some event routine, the routine will be executed from the beginning at the next event.

The task(s) available are dependent on the type **Tasks**.

Limitation for POWER ON, RESET and QSTOP event: The specified event routine cannot be executed if the task program has semantic errors (reference errors etc.). If this is the case, the system will generate an error.

5.4 Specifying regain distances

Maximum distance for a regain movement (the distance from the current robot position to the last executed path). This can be set both for start in manual mode and for start in automatic mode.

A regain movement will begin when program start is ordered and before the program continues with the instruction that was interrupted due to a stop request. If the regain distance exceeds the specified max. distance, an error message will occur.

• Choose **Types: Regain distances.**

The operating modes will be listed, (see Figure 18).

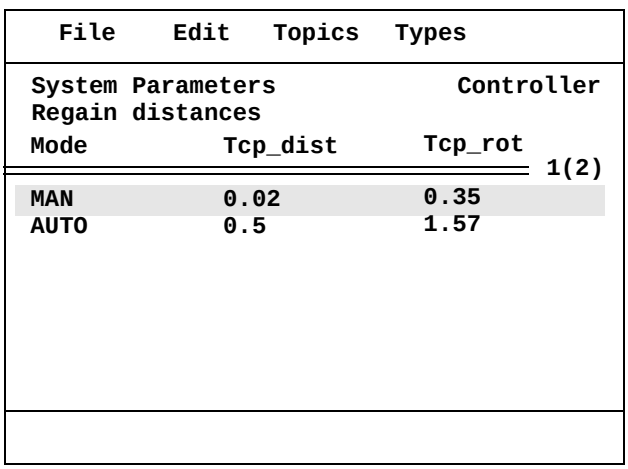


Figure 18 Maximum regain distances.

• Select the operating mode to be changed and press Enter ↵ .

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Mode	AUTO or MAN.
Tcp_dist	The maximum TCP distance (m).
Tcp_rot	The maximum TCP rotation (rad).
Ext_dist	The maximum distance for external axes (m).
Ext_rot	The maximum rotation for external axes (rad).

5.5 System miscellaneous

Changes to any item in this menu will force the system to restart the program handling part of the system at the next warm start. All user programs will be erased and all task modules specified in the configuration will be reloaded.

• Choose **Types: System misc.**

All functions already added will be listed, (see Figure 19).

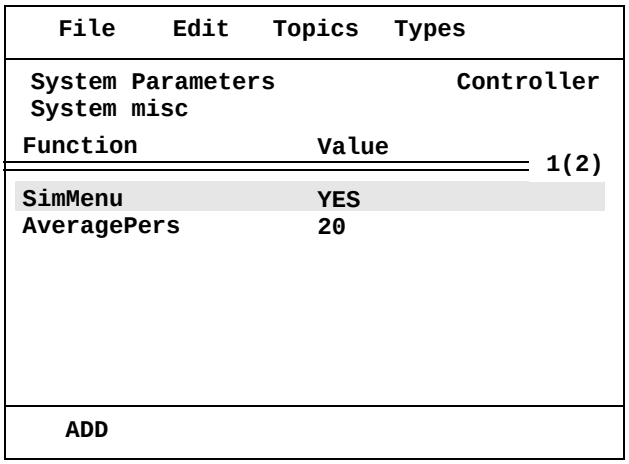


Figure 19 System miscellaneous.

• Mark the function to be changed and change it, or add a new one.

• Press ***OK*** to confirm.

Function	Description
SimMenu	The *WaitTime*, *WaitUntil* and *WaitDI* instructions will generate an alert box in manual mode to make it possible for the user to simulate the instruction and continue to execute the next instruction. If this is set to **NO**, no menu will be generated. **YES** is the default behaviour.
AveragePers	Average size in bytes of one PERSISTENT variable. This setting will affect the maximum number of persistents in the system.

5.6 Automatic loading of modules and programs

System modules and/or normal RAPID program modules can be loaded automatically when the robot is powered on (restarted).

Changes to any item in this menu will force the system to restart the program handling part of the system at the next warm start. All user programs will be erased, and all task modules specified in the configuration will be reloaded.

• Choose **Types: Task modules.**

A list of the files which will be preloaded will be shown, (see Figure 20).

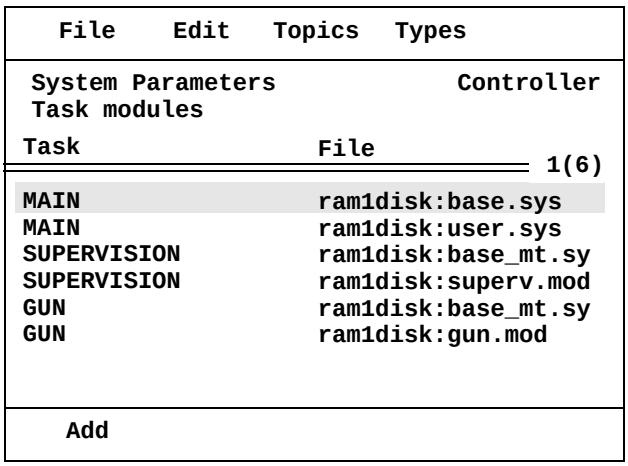


Figure 20 Programs loaded into the system during the warm start sequence.

• Select the item to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Task	The symbolic name of the task to which the module should be loaded. The available task(s) is shown under the type **Tasks**. (See *Defining multitasking* on page 42).
File	A path to the module file. (**Note:** The file must be reachable in every warm start, e.g. ram1disk:base.sys)
Storage	**Built in** or **loaded**. A built in module is not visible, it will not occur in the list of modules and cannot be removed from the program window (see Developer's Manual). Loaded is the default behaviour.
TextResource	If Storage is set to **Built in** it is possible to use a national language for routine names, for example. This parameter should be 0 as English is used for the RAPID language (see Developer's Manual).

Shared If Storage is set to **Built in** it is possible in a multitask system to install the module so it (and all its objects) will be reachable from all tasks. This parameter should then be set to **YES**. These objects are also called *Intertask Objects* (see Developer's Manual). The default behaviour is **NO**.

The files "ram1disk:base.sys" and "ram1disk:user.sys" are predefined and should not be removed, but the contents of "user.sys" may be modified.

The file "ram1disk:base_mt.sys" should always be defined for any additional tasks.

5.7 Defining multitasking

Available when the option *Multitasking* is installed. The various tasks are defined with name, priority and execution behaviour.

Changes to any item in this menu will force the system to restart the program handling part of the system at the next warm start. All user programs will be erased and all task modules specified in the configuration will be reloaded.

• Choose **Types: Tasks.**

All specified tasks will be listed, (see Figure 21).

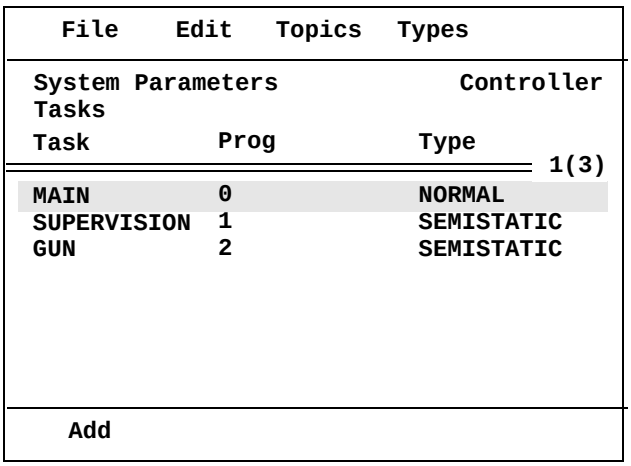


Figure 21 All available tasks.

• Select the task to be changed and press Enter ↵, or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Task	The name of a task. (max 16 characters)
Prog	The program number. Program no. 0 is reserved for the normal robot program which is the only one that may include motion instructions.
Type	Controls the start/stop and system restart behaviour. **NORMAL** - The task will react on START/STOP requests given from the teach pendant or other sources. **STATIC** - The task will be started from the beginning at the first warm start after a cold start, and automatically restarted at the current position at all other warm starts. **SEMISTATIC**- The task will be restarted from the beginning at all warm starts. Program no. 0 must be of type NORMAL. The other tasks should be of type STATIC or SEMISTATIC.
Task in foreground	The name (or program number) of a task that should run in foreground the of this. If “-1” or an empty string ““ is set for this parameter, it will run at the highest level with no other task that could suspend its execution.
Main entry	The name of the start routine. It should be a RAPID routine without any parameters and reachable in this task (only valid for STATIC and SEMISTATIC tasks).
BindRef	This parameter should be set to **NO** if the system is to accept unsolved references in the program while linking a module, or otherwise set to **YES** (default value is **YES**). The parameter must be set to **NO** if the instructions **Load/Erase** are to be used. There will be a runtime error on execution of an unresolved reference.
TrustLevel	TrustLevel handle the system behaviour when a SEMISTATIC or STATIC task is stopped for some reason or not executable. **SysFail** - This is the default behaviour. All NORMAL tasks (normally only the MAIN task) will be stopped. Besides that the system is set to state SYS_FAIL. All jogging and program start orders will be rejected. Only a new warm start resets the system. This should be used when the task has some security supervisions. **SysHalt** - All NORMAL tasks will be stopped. The system is forced to “motors off”. When taking up the system to “motors on” it is possible to jog the robot, but a new attempt to start the program will be rejected. A new warm start will reset the system. **SysStop** - All NORMAL tasks will be stopped but is restartable. Jogging is also possible. **NoSafety** - Only the actual task itself will stop.

If a task is specified as a STATIC or SEMISTATIC type, all modules must be preloaded. See *Automatic loading of modules and programs* on page 41.

6 Topic: TeachPendant

The following parameters are found under the TeachPendant topic:

- Optional packages
- Defining customised file extensions
- Authorising and confirmation of user commands, changing Pass Codes.
- Activation of limited modpos function
- Programmable keys.
- Default running mode settings

• Choose **Topics: Teach Pendant**.

The Most Common instruction pick lists and I/O list are also stored when saving this topic.

6.1 Defining Optional Packages

If several process packages (ArcWare, SpotWare etc.) have been added to the system, it is possible to define which package is to be used for the Program window and the Production window.

• Choose **Topics: TeachPendant**.

• Choose **Types: Optional Package.**

• Press Enter [↵] .

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Use for Program	The name of the process package to be used for Program, or NONE if not used.
Use for Production	The name of the process package to be used for Production, or NONE if not used. (No process package available in this version).

6.2 Defining File Extension

It is possible to add file extensions for RAPID created files, so that they are recognised by any file dialogue.

• Choose **Topics: TeachPendant**.

• Choose **Types: File Extensions.**

• Select the File extension to be changed and press Enter ↵ or add a new one by pressing ***Add***.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the extension (max. 3 characters)
Description	Explains the type of data in the file

6.3 Defining authorisation and confirmation

It is possible to limit the access to certain commands, by using user levels and associated pass codes. This means that a function will not be executed unless you have the correct user level. It is also possible to define that a command will not be performed until it is confirmed.

In the robot there are four (4) user levels:

Operator for functions accessible to all users.
No pass code needed.

Service for functions associated with service.
Pass code needed.

Programmer for functions related to programming and testing.
Pass code needed.

Service & ***Programmer***
For functions needed for both programming and service.
Pass code needed for either Service or Programmer.

A pass code can be up to 8 digits long.

If you try to perform a command and you do not have the correct user level, a dialogue will appear, as shown in Figure 22:

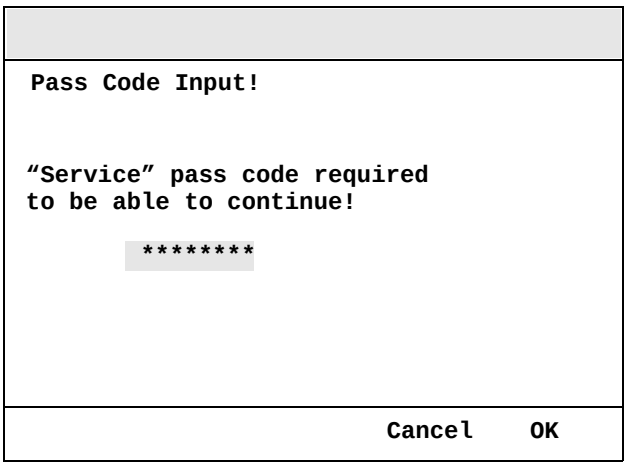


Figure 22 Pass Code Input Dialogue.

• Input your pass code for the correct user level.

• Press ***OK*** to confirm the pass code input.

If the pass code is still not correct, press ***Cancel*** and ask your system administrator for the correct one.

Defining new pass codes

• Choose **Topics: TeachPendant**.

• Choose **Edit: Change Pass Codes**.

• Read the warning message and press ***OK***.

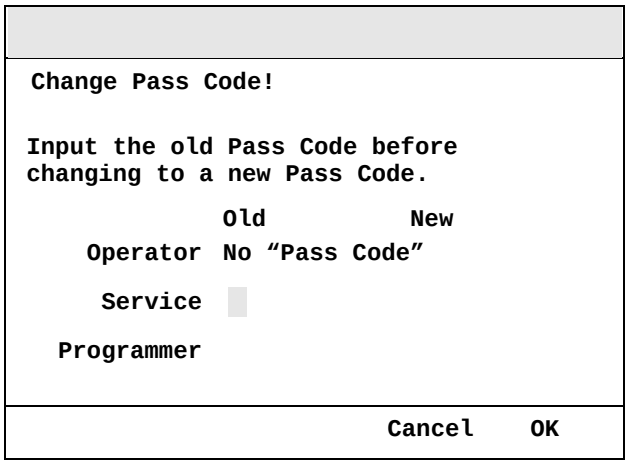


Figure 23 Pass CodeChange Dialogue

• Select the old pass code of the user level to be changed (use the arrow keys Up or Down)

• Input the old pass code (the pass code will not be visible). After installation of the control program, the pass code is *007*.

• Select the new pass code of the user level to be changed.

• Input the new pass code, (the pass code will be visible).

• Press ***OK*** to acknowledge the change of pass code.

• Press Enter ↵ to confirm the updating.

Defining authorisation

To authorise a function:

• Choose **Topics: TeachPendant**.

• Choose from the **Types** menu, the window you want to authorise (names start with **Authorise...**).

All functions that can be authorised will be displayed (e.g. as shown in Figure 24).

• Select the function to change and press Enter ↵.

Parameter	Description
Function	The name of the function to be authorised (cannot be changed).
User Level	Required user level to be able to execute the function, (can be Operator, Service and Programmer).
Confirm	Should the function be confirmed before it is executed? Yes or No.

• To change

- User Level, select parameter ***User Level*** and press Enter ↵.
- Confirm, select parameter ***Confirm*** and press Enter ↵.

• Choose appropriate value and press ***OK***.

• When finished, press ***OK*** to confirm the change.

Authorise SystemParameters

• Choose **Topics: TeachPendant**.

• Choose **Types: Authorise SystemParameters**.

All possible functions will be displayed, as shown in Figure 24.

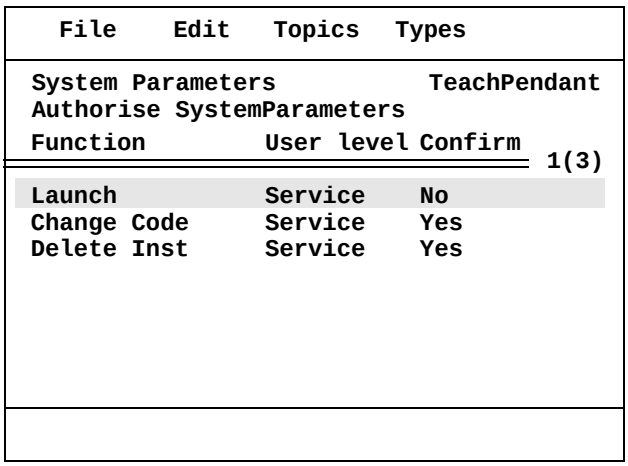


Figure 24 Authorise System Parameters.

Function	Description
Launch	To authorise the opening of the window.
Change Code	To authorise the change of pass codes.
Delete Inst	To authorise the deletion of a parameter.

• To change user level and/or confirm, see *Defining authorisation* on page 47.

Authorise Program

• Choose **Topics: TeachPendant**.

• Choose **Types: Authorise Program.**

All possible functions will be displayed, as shown in Figure 25.

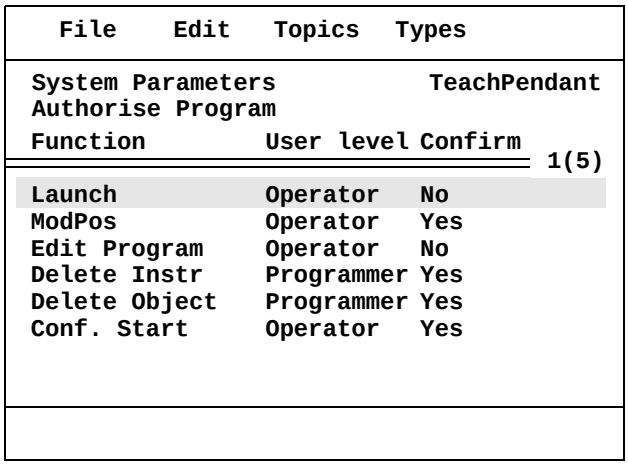


Figure 25 Authorise Program.

Function	Description
Launch	To authorise the opening of the window.
ModPos	To authorise modification of a position
Edit Program	To authorise changing of the program.
Delete Instr	To authorise the deletion of any instruction in a RAPID routine.
Delete Object	To authorise the deletion of any RAPID objects (e.g. routines, modules or data).
Conf. Start	Only confirmation. If set to ***No*** the program execution will always start from the program pointer (PP).

• To change user level and/or confirm, see *Defining authorisation* on page 47.

6.4 Activation of Limited ModPos Function

If the Limit ModPos function is active, only a limited deviation from the original position is allowed, when the ***ModPos*** key is pressed to modify a position. The limited deviation concerns both the linear distance and the orientation.

• Choose **Topics: TeachPendant**.

• Choose **Types: Modify Position**.

Now the current type of ModPos function will be displayed, ***ModPos*** or ***LModPos***.

• Press Enter [↵] .

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Type	The current type of modpos. *ModPos* means that Limit ModPos is deactivated, i.e. any change is accepted. *LModPos* means that Limit ModPos is activated, i.e. the change must be within a limited area.
Tuning In Auto	Tuning Off or On in auto. *On* = tuning functionality enabled in automatic mode. *Off* = tuning functionality disabled in automatic mode.
Mode	The current mode of limited modpos. *Abs* = The limited area is around a fixed original point. *Rel* = The limited area is around the current point and will be moved when you modify the point.
Max Trans	The maximum allowed deviation in mm from the current or original position.
Max Rot	The maximum allowed reorientation in degrees from the current or original position.
Max External Trans	The maximum allowed deviation in mm from the current or original position concerning external linear axes.
Max External Rot	The maximum allowed deviation in degrees from the current or original position concerning external rotational axes.
If Auto	Parameter for automatic activation of Limit ModPos when the operator's key is switched to Auto Mode. *LModPos* means that Limit ModPos is activated when the operator's key is switched to Auto Mode. *As Is* means that ModPos is not changed.

6.5 Programmable keys

There are five keys on the teach pendant which you can define for the purpose of setting outputs and generating signal events, see Figure 26.

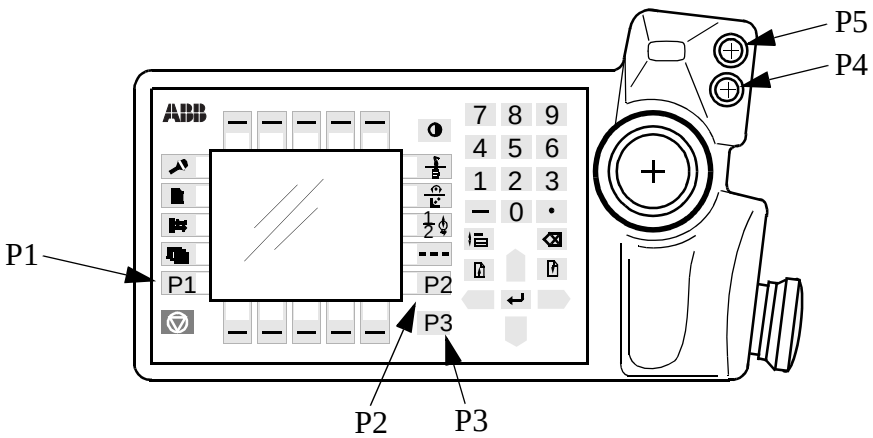


Figure 26 The five programmable keys.

- Choose **Topics: Teach Pendant**.
- Choose **Types: Programmable Keys**.

Now the definition of the keys will be displayed.

• Select the key to be defined and press Enter [↵] .

Parameter	Description
Key	The designation of the key P1-P5
Type	Type of key: **Input**, **Output** or **None** (not activated)
Connection	Name of signal to be chosen.

When Type is selected as Output, the following are also available:

Key Pressed	Defines how the output should be set. Toggle: if the signal value is high (1), it will become low (0) and vice versa. Pulse: a positive pulse (200 ms) is generated. Set1/Set0: either to high (1) or to low (0). Press/Release: the signal will be high (1) as long as the key is depressed. When the key is released, the signal will change to low (0).
Allow in Auto	Defines if the output should be possible to use even in automatic mode. **Yes** allows the key to be operational in both manual and automatic mode. **No** allows the key to be used only in manual mode.

When Type is selected as Input, an event will be generated for this input. This event can be related to:

- System input: The input must then be associated with a system activity, see *Defining input and output signals* on page 12.
- Interrupt: This is defined by the instruction ISignalDI (see RAPID Reference Manual).
- Waiting for input: This is programmed via the instruction WaitDI (but not WaitUntil). (See RAPID Reference Manual).

6.6 Defining Running Mode Settings

The default set-up for the running mode can be defined for the program window (manual mode) and the production window (automatic mode).

• Choose **Topics: TeachPendant**.

• Choose **Types: Running Mode.**

• Press Enter [↵] .

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Operating mode	Manual or Automatic mode
Running Mode	Running mode (Continuous or Cycle) when starting the robot in this mode or switching to this mode.

7 Topic: Manipulator

The Manipulator topic contains parameters associated with motion control in the robot and external axes, e.g.:

- The commutation offset
- The calibration offset
- The working space limits.

• Choose **Topics: Manipulator**.

Do not change "Transm gear ratio" or other kinematic parameters from the teach pendant or a PC. This will affect the safety function *Reduced speed 250 mm/s*.

Regarding available parameters, please note the following:

Some parameters described here require the boot sequence service to be available when making changes, see the Product Manual - *Installation and Commissioning*, chapter Installing the Control Program. It may also be convenient to make a backup and restore it later on, see Chapter 14, Service - *Backup and Restore* in this manual.

7.1 Defining the commutation offset and calibration offset of the motors

These values are generally updated from the Service window. If, however, they are known, they can be specified in the System Parameters window.

• Choose **Topics: Manipulator**.

• Choose **Types: Motor Calib**.

• Select the desired motor and press Enter ↵ .

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the motor, e.g. motor_1 (max. 16 characters).
Calibration offset	The position of the motor (resolver) when it is in the calibration position (in radians).
Cal offset valid	Specifies whether the calibration offset is defined.
Commutator offset	The position of the motor (resolver) when the rotor is in the electrical zero position relative to the stator (in radians).
Com offset valid	Specifies whether the commutation offset is defined.

7.2 Defining the range of movement and calibration position of each axis

- Choose **Topics: Manipulator**.
- Choose **Types: Arm**.
- Select the desired arm (axis) and press Enter ↵.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Name	The name of the arm, e.g. arm_1 (max. 16 characters).
Upper joint bound	The upper joint limit, e.g. +3.139 for axis 1 (in radians). 180^{o} = 3.1416 radians.
Lower joint bound	The lower joint limit, e.g. -3.139 for axis 1 (in radians).
Use check point	The name of a check point (if any). See *Defining arm check point* on page 58.
Use arm load	The name of an arm load (if any). See *Defining arm load* on page 57.
Calibration position	The position of the axis when it was calibrated. If this value is changed, the robot must subsequently be fine-calibrated in the Service window. See the Product Manual.

7.3 Defining supervision level

It is possible to change the default supervision levels if a system needs to be more or less tolerant to external disturbances. A higher tune factor than 1.0 gives a more tolerant robot system and vice versa with a tune factor lower than 1.0. For instance, an increase from the default tune factor 1.0 to factor 2.0, doubles the allowed supervision levels, making the robot system more tolerant to external disturbances. Decreasing the tune factor to 0.5 gives a system which is less tolerant to external disturbances.

Increasing the tune factors can reduce the lifetime of the robot.

- Choose **Topics: Manipulator**.
- Choose **Types: Arm**
- Select the desired arm (axis) and press Enter ↵.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Jam sup trim factor	Tune factor for jam supervision. The factor influences the maximum time allowed at zero speed with maximum torque.
Load sup trim factor	Tune factor for load supervision. The factor influences the maximum time allowed at non-zero speed with maximum torque.
Speed sup trim factor	Tune factor for speed supervision. The factor influences the maximum allowed speed error.
Pos sup trim factor	Tune factor for position supervision. The factor influences the maximum allowed position error.

7.4 Tuning the motion supervision

Motion supervision is the name of a collection of functions for high sensitivity, model-based supervision of the robot's movements. The motion supervision includes functionality for the detection of collisions, jams and incorrect load definitions.

The motion supervision can be tuned through the system parameters. Separate parameters exist for jogging and program execution. **Currently, the names of the system parameters refer to the collision detection. However, they also turn on and off and modify the supervision levels of the load and jam supervisions.**

If the motion supervision is triggered, then do one of the following:

- If the load is incorrectly defined, use the load identification function to define it.
- If the application involves many external process forces, increase the supervision level for jogging and program execution in steps of 30 percent until you no longer receive the error code.
- If the external process forces are only temporary (such as closing a large spotweld gun), then use the instruction MotionSup to raise the supervision level or turn the function off temporarily.
- If everything else fails, turn off the motion supervision.

The motion supervision parameters do not require a restart when modified. Use the following procedure to change the motion supervision system parameters.

- Choose **Topics: Manipulator**.
- Choose **Types: Motion Sup**.
- Select the desired instance (usually IRB) and press Enter [↵].
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Path col detect	Turn the collision detection ON or OFF for program execution (also modifies the load and jam supervisions).
Jog col detect	Turn the collision detection ON or OFF for jogging (also modifies the load and jam supervisions).
Path col detect level	Modifies the supervision level for the collision detection for program execution by the specified percentage value (also modifies the load and jam supervisions). A large percentage value makes the function less sensitive.
Jog col detect level	Modifies the supervision level for the collision detection for jogging by the specified percentage value (also modifies the load and jam supervisions). A large percentage value makes the function less sensitive.

7.5 Defining teach mode speed

When there is a requirement to monitor manual mode with reduced speed lower than 250 mm/s, this can be achieved by changing the teach mode maximum speed.

- Choose **Topics: Manipulator**.
- Choose **Types: Motion system**
- Select the desired system and press Enter ↵.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Teach mode max speed	Maximum allowed speed in manual mode with reduced speed.

7.6 Defining independent motion

Independent motion is a part of Advanced Motion, and the system has to be loaded in service mode in order to be able to change the independent parameters. The control program must be re-installed to activate these parameters. First select the Query mode. When the question *Service/Standard movement parameters?* comes up, choose ***Service.*** See the Product Manual - *Installation and Commissioning.*

For the manipulator, only axis 6 can be used in independent mode. For external axes there are no limits for independent motion.

- Choose **Topics: Manipulator**.
- Choose **Types: Arm**
- Select the desired arm and press Enter ↵.
- Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Independent joint	Flag used to allow independent mode for this axis.

For external axes, the transmission ratio must be defined as normal with the parameter ***Transm. gear ratio,*** but also with its nominator and denominator values.

• Choose **Types: Transmission** and specify the following:

Parameter	Description
Transm. high gear	The integer value of the numerator of the transmission gear ratio. Only used for independent joints.
Transm. low gear	The integer value of the denominators of the transmission gear ratio. Only used for independent joints. Example: For a rotating axis with high gear 100 and low gear 30, has a transmission gear ratio of 100/30=3.333333 .

For both manipulator axes and external axes the working area can be increased up to:

Manipulator	Work area (radians arm side)
IRB 1400	±1885
IRB 2400.1.8	±1885
IRB 2400.1.5	±1571
IRB 4400	±1260
IRB 6400	±943
External axes	±1256637 (motor side)

• Choose **Types: Arm** and specify the arm characteristics for the axis.

Parameter	Description
Positive limit	Upper limit for the axis work area (in radians or metres).
Negative limit	Lower limit for the axis work area (in radians or metres).

7.7 Defining arm load

The performance of the robot will be negatively affected if the arm load is not defined. When more than one load is mounted on one and the same arm, the total weight and the position of the centre of gravity must be calculated.

All loads mounted on the upper arm are related to axis 3, including loads on the rotating part.

• Choose **Topics: Manipulator**.

• Choose **Types: Arm load**.

Parameter	Description
Name	The name of the arm load, e.g. armload_1 (max. 16 characters).
Mass	The mass of the arm load (in kg).
Mass centre x ***Mass centre y*** ***Mass centre z***	The mass centre, specified using the coordinate system of the arm (in m). See the example in Figure 27.

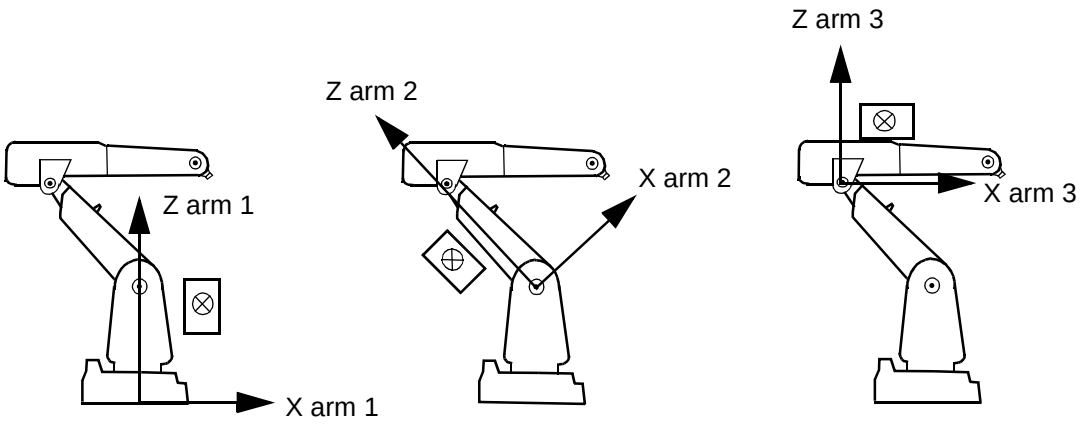


Figure 27 The arm coordinate system for axes 1, 2 and 3.

Now, this arm load must be connected to the current arm (axis):

- Choose **Types: Arm**.
- Select the desired arm and press Enter ↵.
- Select the parameter ***Use customer arm load*** and specify the name of the arm load previously defined.
- Press ***OK*** to confirm.

7.8 Defining arm check point

If an extra load, such as a transformer or a welding-bar roller, is attached to an arm, then a point on this equipment can be defined. In this case, the robot monitors the speed of that point so that it does not exceed 250 mm/s in the manual operating mode (reduced speed).

- Choose **Topics: Manipulator**.
- Choose **Types: Arm check point**.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Name	The name of the check point, e.g. chk_pnt_1 (max. 16 characters).
Position x ***Position y*** ***Position z***	The position of the check point, specified on the basis of the current coordinate system of the arm (in meters). See the example in Figure 28.

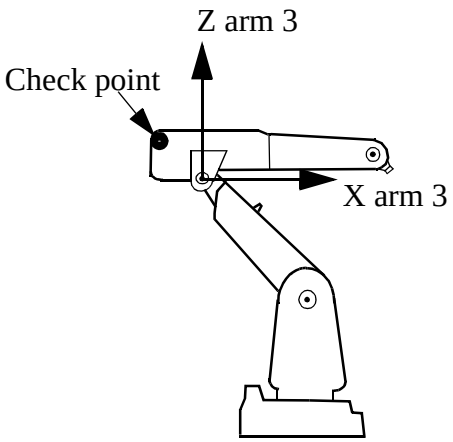


Figure 28 Definition of the check point for arm 3.

Now, this check point must be connected to the current arm (axis):

- Choose **Types: Arm**.
- Select the desired axis and press Enter ↵.
- Select the parameter ***Use check point*** and specify the name of the arm check point previously defined.
- Press ***OK*** to confirm.

7.9 Defining external torque

When external equipment, for instance a cable or a coiled hose, affects any joint significantly, the external torque should be defined using the following formula:

$$T = A + |k \cdot (\theta - \theta_0)|$$

T = external torque [Nm]

A = constant torque [Nm]

k = scale factor for position dependent torque [Nm/rad]

θ_0 = joint position when position dependent torque is zero [rad]

The formula is also illustrated in the figure below.

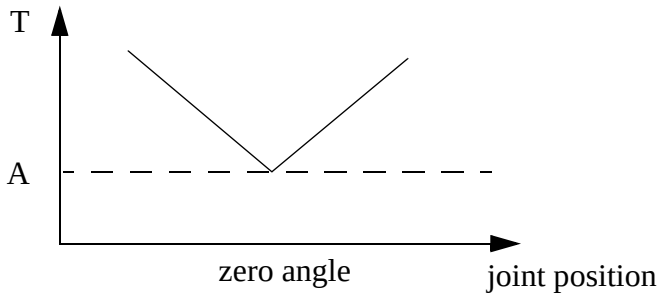


Figure 29 External torque

Example: A coiled hose is mounted and affects joint 6 as follows:

• 0 Nm at 0 degrees

• 5 Nm at 200 degrees

This external torque can be defined using $A = 0$, $\theta_0 = 0$, $k = \frac{5}{200 \cdot (\pi/180)}$.

If the estimated value of a significant external torque is too low, there could be unnecessary path deviations and the manipulator could be damaged. If the estimated value is too high, the performance of the manipulator will be reduced due to restrictive acceleration limits.

• Choose **Topics: Manipulator**.

• Choose **Types: Arm**.

• Select the desired arm and press Enter ↵.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Ext const torque	External constant torque [Nm].
Ext prop torque	Scale factor for position dependent torque [Nm/rad].
Ext torque zero angle	Joint position when position dependent torque is zero [rad].

7.10 Defining friction compensation

Friction compensation can be activated to reduce path errors caused by friction and backlash at low speeds (10 - 200 mm/s).

The friction model is a constant level with a sign opposite to the axis speed direction. *Friction ffw level (Nm)* is the absolute friction level at (low) speeds greater than *Friction ffw ramp (rad/s)* see Figure 30.

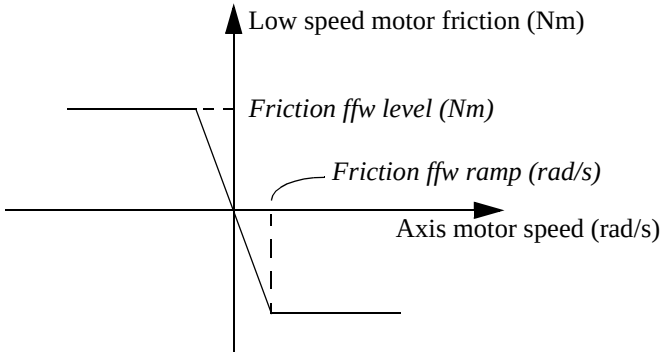


Figure 30 Friction model.

The instruction TuneServo can be used to tune optimal values for each robot axis.

• Choose **Topics: Manipulator**.

• Choose **Types: Control parameters**.

• Select the desired axis and press Enter ↵.

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Friction ffw on	Activates/deactivates friction compensation.
Friction ffw level	Low speed motor friction level [Nm].
Friction ffw ramp	Friction ramp [rad/s].

Tuning procedure

The most straight forward way to tune the friction parameters is to consider one axis at a time. Choose a motion which has a characteristic friction “bump” due to a specific axis changing direction. Turn the friction compensation on for this specific axis and use the default setting for friction level and ramp. Use the TuneServo command to gradually increase the friction level until the bump in the path is removed.

Repeat this procedure for all axes.

7.11 Defining the base coordinate system

Normally the base coordinate system of the robot coincides with the global coordinate system. However, the base coordinate system can be moved relative to the global coordinate system. Please note that the programmed positions are always related to the global coordinate system, and all positions will therefore also be moved, as seen from the robot. Normally this would be defined from the Service window, but if the values are known they can be input from the system parameters.

• Choose **Topics: Manipulator**.

• Choose **Types: Robot**.

• Select the manipulator whose base coordinate system is to be changed.

• Press Enter ↵ .

• Select the desired parameter and change its value.

• Press ***OK*** to confirm.

Parameter	Description
Name	The name of the robot, e.g. robot_1 (max. 16 characters).
Type	Robot type (not to be changed).
Base frame x	The X-coordinate of the base coordinate system's position in relation to the world coordinate system (in metres).
Base frame y	The Y-coordinate of the base coordinate system (in metres).
Base frame z	The Z-coordinate of the base coordinate system (in metres).
Base frame q1-q4	The orientation of the base coordinate system in relation to the world coordinate system (expressed in quaternions q1-q4). Figure 31 illustrates some examples of different values.
Base frame moved by	Specifies whether the robot is to be operated in coordination with a track. See *Defining a track motion with coordinated motion* on page 63.

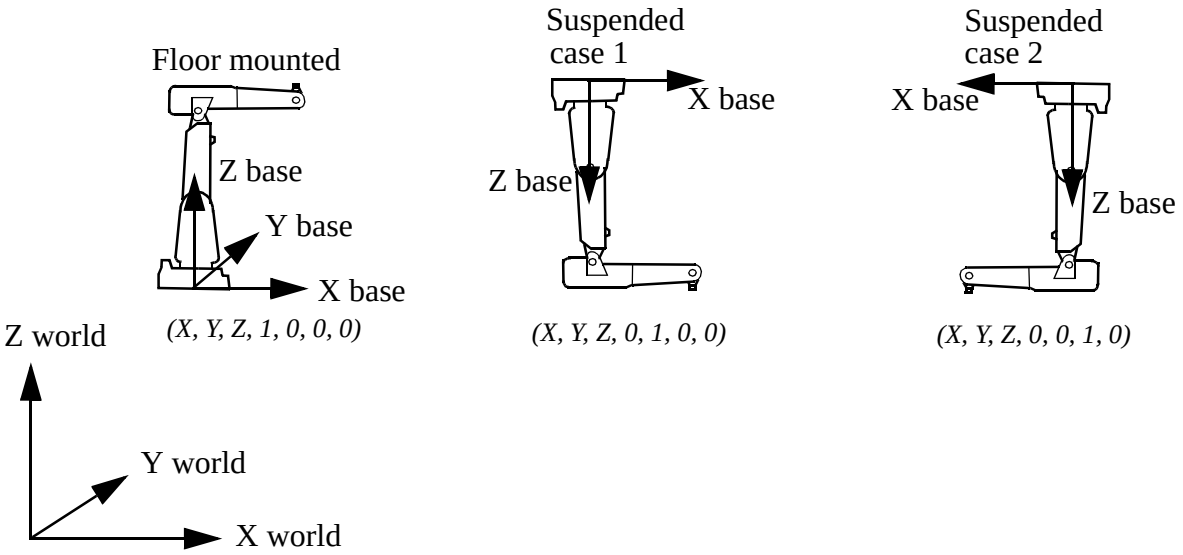


Figure 31 Some examples of definitions of the base coordinate system.

7.12 Defining external manipulators with more than one axis

To achieve the best possible performance from an external manipulator, a set of data, describing its kinematic and dynamic properties (among other things), must be defined. This data cannot be defined in the system parameters, but must be read from a diskette. If no diskette was supplied with the manipulator, the manipulator cannot be coordinated with the robot. It can, however, be defined as a number of separate external axes.

- Read the files that define the manipulator. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**.
- Define the calibration offset, name of the mechanical unit, etc. See *Defining external axes* on page 63.
- Define the base coordinate system as described in *Defining an external mechanical unit coordinated with the robot* on page 63.

7.13 Defining a track motion with coordinated motion

- Define the axis in the usual way. See *Defining external axes* on page 63. Define the base coordinate system of the track motion, from the Service window, for example. See Chapter 10, Calibration.
- Choose **Types: Robot**.
- Select the robot and press Enter ↵.
- Set the parameter ***Base frame moved by*** to the name of the axis (single) that is used by the defined track.

7.14 Defining an external mechanical unit coordinated with the robot

- Define the axis in the usual way. See *Defining external axes* on page 63. Define the base coordinate system of the unit from the Service window, for example. See Chapter 10, Calibration.
- Choose **Types: Mechanical unit**.
- Select the mechanical unit to be coordinated with the robot and press Enter ↵.
- Set the parameter ***User frame moved by*** to the name of the axis (*Single*) that rotates the work object.

7.15 Defining external axes

The configuration files supplied must be used for track motion and for manipulators supplied by ABB. See the enclosed documentation.

On the system diskette *Control Parameters*, which is supplied with the robot, there are a number of predefined setups of external axes. These can be found in the directory EXTAXIS and must be used during installation.

- *Service movement parameters* must be used when defining external axes. The control program must be re-installed to activate these parameters. First select the Query mode. When the question *Service/Standard movement parameters?* comes up, choose ***Service.*** See the Product Manual - *Installation and Commissioning*.
- When a separate transformer and rectifier unit is used to power external axes (drive system installed in a separate cabinet), read in the configuration files for the transformer and rectifier unit in question.

Configuration file	Identification number
EXTAXIS/UTIL/TRAFO1D2	3HAC 0747-1 or 0749-1 or 0750-1
EXTAXIS/UTIL/TRAFO2D2	3HAC 0751-1 or 0752-1 or 0753-1
EXTAXIS/UTIL/TRAFO3D2	3HAC 0754-1 or 0755-1 or 0756-1
EXTAXIS/UTIL/DC0_D2	DSQC 345A
EXTAXIS/UTIL/DC1_D2	DSQC 345B
EXTAXIS/UTIL/DC2_D2	DSQC 345C
EXTAXIS/UTIL/DC3_D2	DSQC 345D
EXTAXIS/UTIL/DC2T_D2	DSQC 358C

The files can be found on the *Control Parameters* diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add new parameters**.

Only one file may be read in for the transformer and rectifier unit respectively, even when more than one external axes is used. In this way the transformer and rectifier unit will be automatically defined as common to all the external axes in the drive system 2.

- Read in the axis-configuration files for the required axes, EXTAXIS/MN_DEF/MNxMyDz where x denotes the measurement node number, y the measuring system and z the drive system connected. The files will be loaded from the *Control Parameters* diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add new parameters**.
  Read in the file that corresponds to the current installation, one file for each axis used.

Rules for connecting external axes to the system (for general information see Product Manual - *Installation and Commissioning/External axes*):

A maximum of 6 external axes may be connected to the system.

- Measuring system connections:
  One extra Serial Measuring board can be connected to measuring system 1. On this, one axis can be physically connected to node 4 on the board.
  Up to four SMBs can be connected to measuring system 2. The measurement boards must be numbered in consecutive order starting with board_1_m2 followed by board_2_m2 etc.
  Each axis in the measurement system must have its own unique node number. The axes can be connected to the measuring boards in an arbitrary way.
- Drive system connections:
  Max. one external axis may be activated in drive system 1.
  If drive system 1.2 is used, drive system 2 must not be used.

Only one configuration file per axis number may be read in.

Configuration files with standard data on Control Parameters

Configuration file	Logical axis	Measuring system		Drive system		
		System*	Node*	System*	Unit position	Node
MN4M1D1	7	1	4(7)**	1	0	2
MN4M1D2	7	1	4(7)**	2	1	2
MN4M1D12	7	2	4(7)**	2	0	2
MN1M2D1	8	2	1	1	0	2
MN1M2D2	8	2	1	2	1	1
MN1M2D12	8	2	1	2	0	1
MN2M2D1	9	2	2	1	0	2
MN2M2D2	9	2	2	2	2	2
MN2M2D12	9	2	2	2	0	2
MN3M2D1	10	2	3	1	0	2
MN3M2D2	10	2	3	2	2	1
MN3M2D12	10	2	3	2	0	1
MN4M2D1	11	2	4	1	0	2
MN4M2D2	11	2	4	2	3	2
MN4M2D12	11	2	4	2	0	2
MN5M2D1	12	2	5	1	0	2
MN5M2D2	12	2	5	2	3	1
MN5M2D12	12	2	5	2	0	1
MN6M2D1	7	2	6	1	0	2
MN6M2D2	7	2	6	2	1	2
MN6M2D12	7	2	6	2	0	2

* Parameter value must not be changed.

** Is connected physically to node 4 but the logical value in the system parameters must be 7.

A mechanical unit is now created for each axis. Its name will be the same as in the file, e.g. *MN4M1D1*.

- Now restart using **File: Restart.** (Do not worry about any error codes, which are caused by drive system parameters that have not yet been updated.)
- Choose **Area: Manipulator**.
- Choose **Types: Mechanical unit**, and specify the following:

Parameter	Description
Name	Name of the unit (max. 7 characters). This name will be used later in the Jogging window and in the program, e.g. when a unit is activated.
Standby state	The unit is controlled and the brakes are released not until the first movement instruction is executed or until it is moved manually. Must not be deactivated in the S4C system.
Activated when starting	Unit is automatically activated when starting.
Deactivate not allowed	Unit must not be deactivated.

• Choose **Types: Single Type** and specify under parameter ***Mechanics***:

TRACK	- Linear track motion
FREE_ROT	- Rotating external axis
EXT_POS	- Used internally
TOOL_ROT	- Used internally

• Choose **Types: Joints** and specify, under parameter ***Logical axis***, the logical number of the axis in the RAPID program. Example: Logical axis 10 will then correspond to the field *eax_d* in a data of the type *robtarget*.

• Choose **Types: Arm** and specify the arm characteristics for the axis.

Parameter	Description
Positive limit	Upper limit for the axis work area (in radians or metres).
Negative limit	Lower limit for the axis work area (in radians or metres).

• Choose **Types: Acc data** and specify the arm performance for the axis.

Parameter	Description
Nominal acceleration	Axis acceleration in radians/s^2. If the value specified is too high, the motor will reach the torque limit.
Nominal deceleration	Axis deceleration in radians/s^2. If the value specified is too high, the motor will reach the torque limit.

• Choose **Types: Transmission** and specify the following:

Parameter	Description
Transm. gear ratio	Gear ratio between motor and axis. Example: For a linear axis, 21.43 denotes that when the axis moves 1 m, the motor rotates 21.43 radians.
Rotation movement	Denotes whether the axis is of the rotating type (***Yes***) or linear type (***No***).
Transm. high gear	The integer value of the numerator of the transmission gear ratio. Only used for independent joints.
Transm. low gear	The integer value of the denominators of the transmission gear ratio. Only used for independent joints. Example: For a rotating axis with high gear 100 and low gear 30, has a transmission gear ratio of 100/30=3.333333.

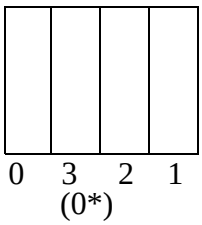
• Choose **Types: Drive unit** and specify the following:

Parameter	Description
Use drive unit type	Current type of drive unit.
Unit position	Denotes the physical location in the cabinet.
Node	Denotes the node on the drive unit to which the axis is physically connected.
Max current (A)	Denotes the max. current of the drive node.
Max thermal current (A)	Denotes the max. thermal current of the drive node.

Data for drive units

Drive unit type	Node	Max. current (A)	Max. thermal current (A)
DSQC 346A	1	3.25	1.6
	2	3.25	1.6
	3	1.5	1.0
DSQC 346B	1	6.7	3.2
	2	3.25	1.6
	3	1.5	1.0
DSQC 346C	1	11.3	5.3
	2	11.3	5.3
	3	6.7	4.0
DSQC 346G	1	29.7	16.5
	2	36.8	20.0
DSQC 358C	2	36.8	20.0

Positions of drive units in the drive system:



* Drive unit position of drive system 12

Figure 32 Drive system as seen from the front of the cabinet.

• Choose **Types: Motor type** and specify the following:

Parameter	Description
Polepairs	Number of pole pairs for the motor, typically 2 or 3.
ke (V/rad/s)	Nominal voltage constant, induced voltage phase to phase.
Max current (A rms)	Max. current without irreversible demagnetisation.
Phase resistance (Ohm)	Stator phase resistance (half the resistance value measured between incoming phases).

Phase inductance (H) Stator phase inductance.

• Choose **Types: Motor_calibration** and define the calibration and commutation offsets as described in *Defining the commutation offset and calibration offset of the motors* on page 53.

Commutation (Normally commutation does not need to be implemented because the motors supplied by ABB are precommutated with the commutation value 1.5708).

Input commutation offset for an uncommutated motor:
1. Ensure that the motor is not affected by large external torques (gravity and friction).
2. Connect a DC power supply to the motor (S-phase +, T-phase -, R-phase not connected).
3. Feed in as large a current as possible with regard to the max. permissible currents of the motor.
4. The motor will now assume the commutation position.
5. Commutate the motor from the Service window.

Calibration offset can be updated by moving the axes to their calibration positions and then fine-calibrating from the Service window.

• Choose **Types: Max operational cycle** and specify the characteristics of the drive system at its maximum usage.

Parameter	Description
Torque abs.max	Max. allowed torque (Nm).
Speed abs.max	Max. allowed motor rotational speed (radians/s).

• Restart using **File: Restart.**

Define activating relays (if any are connected)

The external drive units can be activated via signals from the robot. When a unit is activated, e.g. by choosing the unit in the Jogging window, the output signal is automatically set. A check is made later that the corresponding input signal from the relay is set.

• Define the input and output signals for the relays. See *Defining input and output signals* on page 12. Restart the robot using **File: Restart**, and check that the external axes can be activated from the I/O window.

• Read in a relay configuration file, EXTAXIS/UTIL/RELAY, from the Control Parameters diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add new parameters**. A new relay with the name RELAY_x will now be created.

• Choose **Area: Manipulator**.

• Choose **Types: Relays** and specify the name of the relay and its signal connections.

Parameter	Description
Name	Name of the relay, e.g. relay_track (max. 16 characters). This must be changed when more relays are to be used.
Out signal	Denotes the logical name of the output signal to the relay. The name must be identical (including upper and lower case letters) to the name used for the signal definition.
In signal	Denotes the logical name of the input signal to the relay. The name must be identical (including upper and lower case letters) to the name used for the signal definition.
Inverted input	Denotes whether or not an inverted input is to be used to the relay.

- Choose **Types: Mechanical unit** and specify under the parameter ***Use act.relay*** the name of the activating relay.
- When more than one activating relay is used, read in a new relay file (file RELAY) and repeat the procedure above.

Define brake relays (if any are connected)

If the external mechanical units are equipped with brakes, these will be automatically activated when the unit is deactivated or when the robot system assumes the MOTORS OFF state. They will also be activated when the axes have been stationary for a certain time in the MOTORS ON state.

- Defining the input and output signals for the relays. See *Defining input and output signals* on page 12. Start the robot with **File: Restart**, and check that the brakes can be activated from the I/O window.
- Read in a relay configuration file, EXTAXIS/UTIL/RELAY, from the Control Parameters diskette. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add new parameters**. A new relay with the name RELAY_x is now be created.
- Choose **Area: Manipulator**.
- Choose **Types: Relays** and specify the name of the relay and its signal connections. See *Define activating relays (if any are connected)* on page 68.
- Choose **Types: Mechanical unit** and specify, under the parameter ***Use brake relay,*** the name of the activating relay.

The time after which the regulator will cease to brake and let the mechanical brakes themselves brake, can be changed by choosing **Types: Brakes**.

Parameter	Description
Control time delay	Time delay (s) that starts when the speed < *Control speed limit* and before the regulator should be turned off
Control speed limit	Speed (as a % of max. speed) at which the time delay starts.

When more that one brake relay is used, read in a new relay file (file RELAY) and

repeat the procedure above.

• Restart using **File: Restart.**

Tuning the axes

The servo control method can be adjusted to achieve the best possible performance.

• For an uncalibrated axis, choose **Types: Uncal control master 0** and specify the tuning values for ***Kp, Kv*** and ***Ti*** that give good performance. See below under the description for **Lag control master 0** for an explanation of these parameters.

• For a calibrated axis, choose **Types: Lag control master 0** and specify the values that give good performance.

Parameter	Description
Feedforward mode	Possible options are ***No, Spd*** or ***Trq***. Each one corresponds to a different mode of the controller as described below: ***No:*** This is the simplest configuration. The controller is driven by the position error (lag). Because a relatively large lag is needed to move the axis, the position error can be large. ***Spd:*** In this configuration the controller receives information about the desired speed of the axis. As a result, the position lag is greatly reduced compared to the ***No*** configuration. For this reason, ***Spd*** **is the recommended configuration.** ***Trq:*** In this configuration the controller uses the desired speed and acceleration of the axis to calculate the desired motor torque. This requires knowledge of the mass moment of inertia of the axis, which must be supplied by the user. For this reason this configuration is more difficult to tune. It is only recommended for experienced users.
Delay	This parameter should be left at its default value.
Bandwidth	This parameter should be left at its default value.
Df	Dynamic factor. This parameter is only available in the ***Trq*** configuration. It can be used to damp oscillations of the axis due to mechanical resonance. Initially ***Df*** should be left at its default value. It can be adjusted once the other controller parameters have been fixed.
Dw	This parameter should be left at its default value.
Inertia	Total mass moment of inertia.
Kp, gain pos loop	The amplification of the position control, e.g. 15. A high value will give a stiff axis that quickly assumes its new position. The value should be large without inducing overshoot in the position or oscillations of the axis.
Kv, gain speed loop	The amplification of the velocity control, e.g. 2. A high value gives better high frequency stiffness, better response speed and low overshoot. If the value is too high the axis will vibrate.

Kv controls the amount of damping for the axis and is the most limiting of the parameters. A poor value of ***Kv*** will limit ***Kp*** and ***Ti,*** and the axis will not be fully utilised.

Ti, I-time speed loop The integration interval constant of the velocity control, e.g. 0.2. A low value gives low steady state error and better path following. If ***Ti*** is too small the axis will overshoot and the response will be oscillatory.

Tuning strategies for Lag Control Master 0

*Specifying the **Inertia***

If the ***No*** or ***Spd*** configuration is used, the parameter ***Inertia*** should be set to 0. If the ***Trq*** configuration is used, the total mass moment of inertia should be calculated and entered under ***Inertia.*** This is only recommended for experienced users. The inertia is given by:

$$Inertia = Inertia_{Axis}/(Transm\ gear\ ratio)^2 + Inertia_{Motor} + Inertia_{Break}.$$

*Initial tuning of **Kp, Kv** and **Ti***

Set ***Kp*** to 5.

- Select ***Ti*** based on the mass moment of inertia of the external axis. ***Ti*** should be in the range from 0.1 for very light axes (J = 0.3 kgm^2) to 0.5 for the heaviest axes (J = 12 kgm^2). A typical value is 0.3.
- Increase ***Kv*** to its highest value until the axis starts to vibrate/oscillate or a clear vibration can be heard from the axis, either during motion or when stationary. The axis velocity supervision may also indicate speed failure. When you reach the unstable point, divide ***Kv*** by 2.
- Increase ***Kp*** in increments of 0.5 for the fastest response time, until the first signs of overshooting are observed. Then subtract 1 from ***Kp***. If you observe overshooting at a later time, reduce ***Kp*** to an even lower value.

Using a printer

- Connect the printer channels to the test outputs in the cabinet. The outputs are marked 1 and 2 with a common zero point ground. Voltage level ±10V. (Required: a two channel printer, chart recorder, 25-135 mm/s, e.g. Brush 220.) Inputs:
  - X5: TSTOUT1
  - X6: TSTOUT2
  - X7: 0V
  - X8: 0V
- Make sure that the external axis is commutated and calibrated. Any position may be defined as the calibration position.

- Tune the axis so it may be jogged without stopping due to speed or torque supervision. Keep the same values of ***Kp*** and ***Kv*** as above but multiply ***Ti*** by a factor of 2. ***Ti*** will be re-tuned last in the procedure.
- Program a back-and-forth motion of the external axes with test signals enabled. For the final tuning of the control parameters of the axis it is convenient to use the *TuneServo* command. The following program may be used as an example (STN1 is the name of the external axis):

```
PROC main()
   ActUnit STN1;
   TestSign 1,speed STN1,1,1,0;
   TestSign 2,torque_ref,STN1,1,8,0;
   TuneServo STN1,1,TuneValue\Type:=TUNE_KP;
   TuneServo STN1,1,TuneValue\Type:=TUNE_KV;
   TuneServo STN1,1,TuneValue\Type:=TUNE_TI;
   FOR i FROM 1 TO 10 DO
      MoveJ t1,v_tune,fine,too10;
      MoveJ t2,v_tune,fine,too10;
   ENDFOR
   DeactUnit STN1;
ENDPROC
```

The velocity data and test positions can be modified depending to the value that is to be tuned.

TestSign *Output, SignalId, MechUnit, Axis, Scale, Stime*

Output Data type: *num*
Selection of test output, acceptable values are 1 and 2.

SignalId Data type: *testsignal*
Name of the test signal.

MechUnit Data type: *mecunit*
Mechanical unit for which test signal is required.

Axis Data type: *num*
Axis number.

Scale Data type: *num*
Scaling factor. Acceptable values are 1, 2, 4, 8, 16, etc.

Stime Data type: *num*
Sampling time in seconds. The test output is updated with a new value at each sampling.
The value 1 denotes one update every second.
The value 0 denotes updating as often as possible.
The value 0.01 denotes 100 updates per second.

*Tuning the **Nominal acceleration** and **deceleration***

If an axis has a variable moment of inertia, ***Nominal acceleration*** and ***Nominal deceleration*** should be tuned with the maximum inertia.

If gravity has an influence on the axis, then ***Nominal acceleration*** should be tuned with a motion accelerating upwards against gravity. ***Nominal deceleration*** should be tuned with a stopping motion (deceleration) while moving downwards with gravity.

Program two test points for acceleration and two test points for deceleration with the following requirements:

- Velocity: Choose a velocity that is approximately 50% of the maximum speed of the external axis, i.e. speed test signal of approximately 2.5 V.
- Distance: The distance should be chosen to ensure that the axis stabilises at the programmed velocity before deceleration.

1. Using the chart recorder, record the values of *speed* and *torque_ref* for the external axis.

2. Use test positions for ***Nominal acceleration***. Run the motion and check the value of *torque_ref* for the torque limit. Adjust the value of ***Nominal acceleration*** upwards or downwards in increments of 0.5 until the *torque_ref* signal shows that the axis does not reach the torque limit. Reduce the final value by 10% to allow for variations in the mechanical system over a period of time.

3. Use test positions for ***Nominal deceleration***. Run the motion and check the value of *torque_ref* for the torque limit. Adjust the value of ***Nominal deceleration*** upwards or downwards in increments of 0.5 until the *torque_ref* signal shows that the axis does not reach the torque limit. Reduce the final value by 10% to allow for variations in the mechanical system over a period of time.

*Final tuning of **Kp**, **Kv** and **Ti***

If the axis has a variable moment of inertia, ***Kp*** and ***Kv*** and ***Ti*** should be tuned with the maximum value for the moment of inertia.

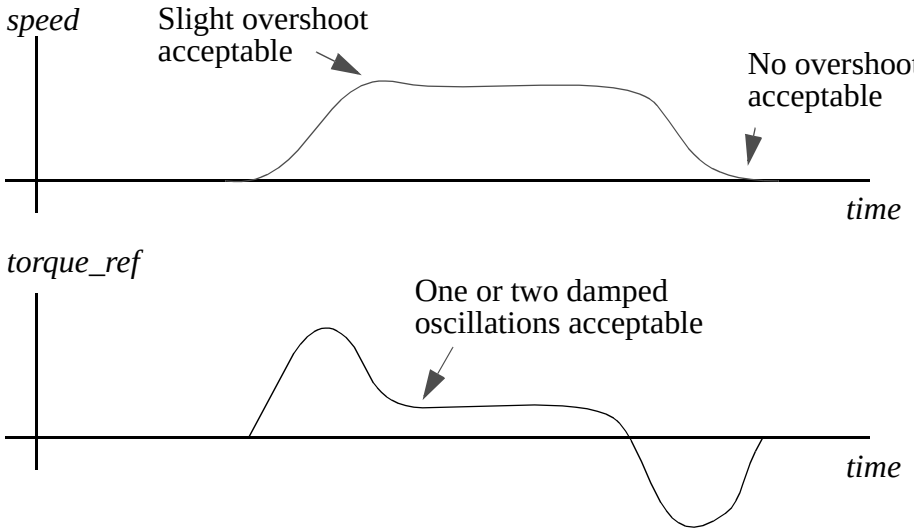
Program two test points with the following requirements:

Velocity: Choose a velocity that is approximately 25% of the maximum speed of the external axis. The speed must be low enough to guarantee that the axis does not encounter the current limit but high enough to prevent friction from affecting the result.

Distance: The distance should be chosen to ensure that the axis stabilises at the programmed velocity before deceleration starts.

- Using the chart recorder, record the values of *speed* and *torque_ref* for the external axis.
- Increase the *TuneValue* for ***Kv*** in steps of 5% and observe *torque_ref*. Stop when the axis starts to vibrate. Divide the *TuneValue* by 2 and run the axis again, while observing *torque_ref*. There should be at most one or two damped oscillations after the acceleration stage. If *torque_ref* oscillates more than this, then decrease its value somewhat. ***Kv*** is a critical parameter. A large value will result in a stiff axis and a fast response. If ***Kv*** is too small ***Kp*** will also be limited, resulting in an under-utilised axis.

• Increase the *TuneValue* for ***Kp*** slowly until the following speed and *torque_ref* profiles are achieved:



The position error (lag) is inversely proportional to ***Kp***. Thus a large value for ***Kp*** is desirable.

• Adjust the *TuneValue* for ***Ti*** downwards in steps of 10% until the effect can be seen on the chart recordings of speed as an increased overshoot. Increase the tuning factor by 5 to 10% to remove the effect.

• Calculate the final values of ***Kp, Kv*** and ***Ti*** by multiplying the value entered under **Lag control master 0** by the *TuneValue* divided by 100. Enter these new values for ***Kp, Kv*** and ***Ti*** under **Lag control master 0.**

Tuning of ***Df***

The dynamic factor (***Df***) can be used to damp oscillations of the axis due to mechanical resonance. This parameter is only available in the ***Trq*** configuration. For most applications it is not necessary to adjust ***Df***. If however, the *torque_ref* signal is oscillatory due to mechanical resonance, then the performance of the axis can be improved by adjusting ***Df.*** To do this, measure the distance between resonance peaks (in mm) on the plot of the *torque_ref* signal. A rough estimate of ***Df*** can be obtained by dividing this value by the chart recorder speed (in mm/sec):

$$Df = \frac{\textit{chart recorder speed}}{\textit{distance between resonance peaks}}$$

Df should be in the range 3 to 25. To tune ***Df,*** program short back-and-forth motions of the axis at maximum speed. The axis should not be allowed to reach full speed before deceleration. Use the TUNE_DF argument of the *TuneServo* command to adjust ***Df*** and examine the *torque_ref* signal. Adjust ***Df*** until the oscillations in the *torque_ref* signal are damped out.

7.16 Activate forced gain control for an external axis

When activating forced gain control for an external axis, two Types under Manipulator must be considered. In **Lag control master 0**, you can decide which axes should have forced gain control and in **Supervision**, you can decide which axes should affect forced gain control. All the axes that affect forced gain control must be within a certain positional range from the end point before the forced gain control is enabled. This positional range is also specified in **Supervision**. To activate forced gain control for an external axis the system needs to be booted with the service option.

- Choose **Topics: Manipulator**.
- Choose **Types: Lag control master 0**.
- Select the lag control master corresponding to the external axis.
- Press Enter ↵.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Forced control active	Determines whether forced gain control is active for this joint. If set to Yes, ***Affects forced ctrl*** in Supervision should normally also be set to Yes for this joint (see below).
Forced factor for Kp	The forced factor for Kp, if forced gain control is active.
Forced factor for Ki	The forced factor for Ki, if forced gain control is active.
Rise time for Kp	The rise time for forced Kp.

The default values for the forced factors and rise time are recommended but can be changed if necessary.

- Choose **Topics: Manipulator**.
- Choose **Types: Supervision**.
- Select the supervision corresponding to the axis that should/should not affect forced gain control.
- Press Enter ↵.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Affects forced ctrl	Determines whether this joint affects forced gain control.
Forced on pos limit	The upper position limit for forced gain control.
Forced off pos limit	The lower position limit for forced gain control.

The positions are expressed in radians for the motor. Keep ***Forced off pos limit*** = 0 unless the forced factors cause problems, such as noise from the motors, when in position.

Do not change supervision for the robot axes. This can reduce the service life of the robot and impair its performance.

7.17 Activate notch filter for an external axis

This is used only in arc welding applications when a variation in external axis speed affects the welding process. This problem sometimes occurs when both coordinated interpolation and weaving are used. The frequency of the speed variation is typically 2 times the weaving frequency. The notch filter will prevent the external axis from oscillating at the weave motion frequency.

There are two ways of using the notch filter. One way is to lock the notch filter to one specific frequency specified in the parameter ***Notch filter frequency***. The other way is to activate ***Notch auto mode***, which means that the notch filter will automatically adjust to the current weave frequency.

A notch filter should not be used together with Rapid Weave. In ***Notch auto mode*** this is handled automatically and the notch filter will work as if there was no weaving at all. To activate ***Notch auto mode*** the system needs to be booted with the Service option.

- Choose **Topics: Manipulator**.
- Choose **Types: Lag control master 0**.
- Select the lag control master corresponding to the external axis.
- Press Enter ↵.
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
Notch filter activated	Yes (if activated), No (otherwise)
Notch filter frequency	Frequency of speed variation. Used when ***Notch auto mode*** is set to No. Typical value: $\frac{2 \times \text{Weld speed}}{\text{Weave length}}$
Notch filter width	Width of notch filter. A higher value increases the width but can also have a negative effect on the performance (response) of the external axis. Recommended value: 0.2.
Notch auto mode	If Yes, the notch filter frequency will automatically adjust to the weave frequency according to the formula $\frac{2 \times \text{Weld speed}}{\text{Weave length}} \cdot$
auto no weave freq	This frequency is used in the notch filter when ***Notch auto mode*** is set to Yes, but weaving is not being used or Rapid Weave is being used. **Note: The default value should only be changed by advanced programmers.**
auto min freq	The minimum notch filter frequency when ***Notch auto mode*** is set to Yes. **Note: The default value should only be changed by advanced programmers**.

auto max rel change	Factor that sets the maximum instant change in the notch filter when ***Notch auto mode*** is set to Yes. **Note: The default value should only be changed by advanced programmers.**

7.18 Soft servo for external axis

Soft servo can be activated for external axis which are configured with **Lag control master 0**. The behaviour of movements with the soft servo activated is described in the RAPID Reference Manual - *Motion and I/O Principles*.

There are four system parameters to consider when the soft servo is used for an external axis. The parameters are set to default values but can be changed if the system is booted in service mode:

- Choose **Topics: Manipulator**.
- Choose **Types: Lag control master 0**.
- Select the lag control master corresponding to the external axis.
- Press Enter [↵].
- Select the desired parameter and change its value.
- Press ***OK*** to confirm.

Parameter	Description
K soft max factor	Determines the value of the product **Kp*Kv** when the soft servo is used with softness 0%. **K soft max factor** should be in the range 0.1 - 2.0 (default 1.0). When the soft servo is activated with 0% softness, the control parameters **Kp** and **Kv** will be tuned such that **Kp*Kv** = $(\mathbf{Kp*Kv})_{\mathbf{normal}}$***K soft max factor**, where $\mathbf{(Kp*Kv)_{normal}}$ is the product of **Kp** and **Kv** during normal operation
K soft min factor	Determines the value of the product **Kp*Kv** if the soft servo is used with softness 100%. **K soft min factor** should be in the range 0.001 - 0.1 (default 0.01). When the soft servo is activated with 100% softness, the control parameters **Kp** and **Kv** are tuned such that **Kp*Kv** = $(\mathbf{Kp*Kv})_{\mathbf{normal}}$***K soft min factor.**
Kp/Kv ratio factor	Factor used to alter the **Kp/Kv** ratio during soft servo. **Kp/Kv ratio factor** should be in the range 0.1 - 1.0 (default 1.0). In soft servo mode, **Kp** and **Kv** are tuned such that **Kp/Kv** = $\mathbf{(Kp/Kv)_{normal}}$ *** Kp/Kv ratio factor**.
Ramp time	Default time for activation of the soft servo. The default value is 0.5 s.

When the soft servo is activated with an arbitrary softness (0% - 100%), **Kp** and **Kv** are tuned such that the product **Kp*Kv** is equal to a interpolated value between the max value, $(\mathbf{Kp*Kv})_{\mathbf{normal}}$***K soft max factor**, and the min value, $\mathbf{(Kp*Kv)_{normal}}$***K soft min factor**.

Trimming of the soft servo parameters

In most applications these parameters do not have to be trimmed and can be left at their default values.

K soft max factor determines, together with **Kp** and **Kv,** the softness at 0%. If a softness of 0% yields too stiff a servo, the factor should be reduced, and it should be increased if 0% softness gives a servo which is too soft.

K soft min factor determines, together with **Kp** and **Kv,** the softness at 100%. If a softness of 100% gives a too soft servo, this factor should be increased, and it should be reduced if 100% gives too stiff a servo. A tuning algorithm can be:

- Determine a maximum axis movement for which the axis should not move with softness 100%. Such a movement can be 0.1 rad for a rotating axis.
- Determine a minimum axis movement for which the axis should move with softness 100%. Such a movement can be 0.2 rad for a rotating axis.
- Activate the soft servo with softness 100% and perform the two movements
- If the axis moves for both movements, the axis is too stiff and the minimum factor should be reduced. If the axis does not move for any movement, the axis is too soft and the minimum factor should be increased.
- Repeat the last two actions until the axis does not move for the smaller movement but does move for the bigger movement.

The movements in the trim procedure should be done close to the point where the soft servo is activated, to minimize the risk of the axis collapsing.

Kp/Kv ratio factor determines the stability margin for the axis. A value less than 1.0 increases the stability. It not possible to set this parameter to a value larger than 1.0 since the stability of the axis would be jeopardized.

If **Ramp time** is changed, the duration of the activation and deactivation phase will change. A short ramp time can result in snatch of the axis at activation.

7.19 Defining the joystick directions for the robot and external manipulator

These values are available under the System Parameters window.

- Load the jog direction templates file for the mechanical units. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**. The jog direction template files can be found on the CONTROLLER PARAMETER diskette under directory JOGDIR.

File ROBOT_M.CFG is intended for the robot. Loading this file will define the parameter instance name strings “linear_jog_m”, “reorient_jog_m”,and “joint_m”.

File ROBOT_E.CFG is intended for the external manipulator. Loading this file will define the parameter instance name string “joint_e”. When more than one external manipulator is used, each external robot file should have its unique name, e.g. ROBOT_E1.CFG, ROBOT_E2.CFG. Also the JOG_JOINT names in each file should be changed from joint_e to joint_e1, joint_e2, respectively

If joystick directions for several mechanical units are to be defined, the template files for all the units can be loaded here. When the file is loaded perform a **Restart**.

- Choose **Topics: Manipulator**.
- Choose **Types 1: Robot**.
- Select robot unit.
- Press Enter ↵.
- Select the desired parameters (Linear jog dir, Reorient jog dir, Joint jog dir) to be changed. Selecting the “default” type restores factory default jog directions.

Do not perform the Restart requested here.

- Repeat for all robots.
- Press ***OK*** to confirm.
- Choose **Types 2: For each selected parameter type above.**
- Select the desired parameter type.
- Edit the elements.
- Press ***OK*** to confirm.
- Perform a **Restart** to activate the modifications to the joystick directions. When joystick directions for both robots and single axes are to be modified, it is not necessary to perform the restart until all the modifications have been made.

Linear jogging

- For each coordinate system, specify the desired axis direction for the respective joystick movements as shown in Figure 33 (±1 for x-axis, ±2 for y-axis and ±3 for z-axis). The signs denote a positive or negative direction of the motion for a positive joystick deflection.

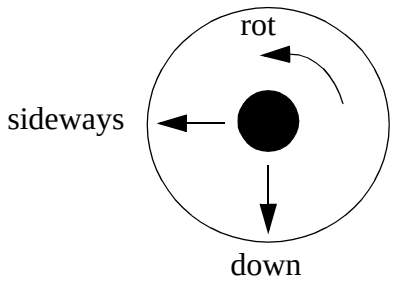


Figure 33 Joystick deflections

Parameters	Coordinate system
World down/rot/side	World coordinate system
Base down/rot/side	Base coordinate system
Tool down/rot/side	Tool coordinate system
Wobj down/rot/side	Object coordinate system
St tool world down/rot/side	World coordinate system when using stationary tool

St tool base down/rot/side	Base coordinate system when using stationary tool
St tool tool down/rot/side	Tool coordinate system when using stationary tool
St tool wobj down/rot/side	Object coordinate system when using stationary tool

Reorientation jogging

• For each coordinate system, specify the desired direction of rotation for the respective joystick movements as shown in Figure 33 (±1 for rotation around the x-axis, ±2 for y-axis and ±3 for z-axis).
The signs denote a positive or negative direction of the motion for a positive joystick deflection.

Parameters	Coordinate system
World down/rot/side	World coordinate system
Base down/rot/side	Base coordinate system
Tool down/rot/side	Tool coordinate system
Wobj down/rot/side	Object coordinate system
St tool world down/rot/side	World coordinate system when using stationary tool
St tool base down/rot/side	Base coordinate system when using stationary tool
St tool tool down/rot/side	Tool coordinate system when using stationary tool
St tool wobj down/rot/side	Object coordinate system when using stationary tool

Axis-by-axis jogging

• Specify the desired axis direction for the respective joystick movements as shown in Figure 33 (±1 for axes 1 and 4, ±2 for axes 2 and 5, ±3 for axes 3 and 6).
The signs denote a positive or negative direction of the motion for a positive joystick deflection.

Parameters	Description
Group 1 down/rot/side	Axis (joint) 1, 2 and 3
Group 2 down/rot/side	Axis (joint) 4, 5 and 6

7.20 Defining the joystick directions for a single external axis

These values are available under the System Parameters window.

• Load the jog direction template file SINGLE.CFG for single external axes. See *Loading parameters from a diskette or some other mass storage device* on page 7. Use the command **File: Add New parameters**. The file can be found on the CONTROLLER PARAMETER diskette under directory JOGDIR.

Loading this file will define the parameter instance ??? name strings “x1”,“x2”, “x3”,“x4”,“x5”, and “x6”.

When joystick directions for several mechanical units are to be defined, the template files for all the units can be loaded here. When the file has been loaded, perform a **Restart**.

- Choose **Topics: Manipulator.**
- Choose **Types 1: Single.**
- Select single.
- Press Enter ↵.
- Select the desired parameters (Single jog dir) to be changed. Selecting the "default" type restores factory default jog directions.

Do not perform the Restart requested here.

- Repeat for all singles.
- Press ***OK*** to confirm.
- Choose **Types 2: Single jog.**
- Select the desired parameter type.
- Edit the elements.
- Press ***OK*** to confirm.
- Perform a **Restart** to activate the modifications to the joystick directions. When joystick directions for both robots and single axes are to be modified, it is not necessary to perform the restart until all the modifications have been made.

Single axis (joint) jogging

Parameters	Description
Joint	The attachment of a single axis of a mechanical unit to a joystick deflection. +/- 1 gives attachment to downways joystick deflection. +/- 2 gives attachment to rotating joystick deflection. +/- 3 gives attachment to sideways joystick deflection.

7.21 Defining kinematic parameters for general kinematics

It is possible to use general kinematics for both master robot and external robots. Definition is not possible via the teach pendant, PC editing of the MOC.cfg file is necessary

Denavit-Hartenberg notation according to John J. Craig, Introduction to Robotics, Mechanics & Control, (Addison-Wesley 1986) is used.

MASTER ROBOT

The following data needs to be defined

• Under **ROBOT_TYPE**

- pos_tol_x, pos_tol_y, pos_tol_z (in meters) (Accepted position error in iterative inverse kinematics procedure)
- rot_tol_x, rot_tol_y, rot_tol_z (in radians) (Accepted orientation error in iterative inverse kinematics procedure)
- base_pose_rot_u0, base_pose_rot_u1, base_pose_rot_u2, base_pose_rot_u3 (Rotation between user defined robot base and internal base according to Denavit - Hartenberg definition)
- no_of_joints = 6
- type **GEN_KIN0** For robot with no wrist axes
- type **GEN_KIN1** For robot with one wrist axis
- type **GEN_KIN2** For robot with two wrist axes

• Under **ARM**

• For each arm of the robot in question

- **rotating_move** if rotating axes, excluded otherwise

• Under **ARM_TYPE**

• For each arm of the robot in question

- **length** (a according to Craigh's definition)
- **offset_x** = 0
- **offset_y** = 0
- **theta_home_position** (theta according to Craigh's definition)
- **offset_z** (d according to Craigh's definition)
- **attitude** (alpha according to Craigh's definition)

Structures with less than 6 axes require a LOCKED definition for all references to the inactive axes.

EXTERNAL ROBOT

The following data needs to be defined

• Under **ROBOT_TYPE**

- base_pose_rot_u0, base_pose_rot_u1, base_pose_rot_u2, base_pose_rot_u3 (Rotation between user defined robot base and internal base according to Denavit - Hartenberg definition)
- no_of_joints = highest joint number
- type **GEN_KIN**

• Under **ARM**

• For each arm of the external robot in question

- **rotating_move** if rotating axes, excluded otherwise

• Under **ARM_TYPE**

• For each arm of the external robot in question

- **length** (a according to Craigh's definition)
- **offset_x** = 0
- **offset_y** = 0
- **theta_home_position** (theta according to Craigh's definition)
- **offset_z** (d according to Craigh's definition)
- **attitude** (alpha according to Craigh's definition)

Structures with less than 6 axes require a LOCKED definition for all references to the inactive axes.

7.22 Servo parameters

Adjustment of the servo to control slow external equipment and the use of brakes when the robot is waiting for movement.

Event preset time

Event preset time is used to delay the robot to make it possible to activate/control external equipment up to 0.5 seconds before the robot runs through the position.

Up to about 70 ms, there is no need to adjust **Event preset time** when the servo has an internal lag. But if a longer adjustment is needed then set **Event preset time** to the longest time wanted.

Example, the parameter **EquipLag** in the **TriggEquip** is set to 200 ms below. If this is the longest time, set **Event preset time** to 200 ms.

```
TriggEquip gunon, 10, 0.2 \Op:=gun, 1;
TriggL p1, v500, gunon, z50, gun1;
```

• Choose **Topics: Manipulator**.

• Choose **Types: motion system**.

• Press Enter [↵].

• Select **Event preset time** and change its value.

• Press ***OK*** to confirm.

Remember that when using **Event preset time,** the start of the robot will be delayed and the performance of weldguide and conveyor will be decreased.

Brake on time

Brake on time will define the time from when the robot stops to activation of the mechanical brakes. This time should be kept high to maintain the reliability of the servo at high level.

7.23 CPU Optimization

In some demanding applications, CPU load problems may occur, causing errors such as “50082 Deceleration too long” or “50024 Corner path failure”. The parameters described below can be used to reduce these problems.

Path resolution

Path resolution corresponds in some sense to the distance between two points in the path. Increasing **Path resolution** means increasing that distance which leads to a *decrease* in the resolution of the path!

Increasing **Path resolution** is a way to deal with robot installations that have external axes with long deceleration times. In such applications the warning “50082 Deceleration too long” will be reported, simultaneously generating a quick-stop. The path resolution parameter will then need to be increased until the problem disappears.

The need for tuning the path resolution parameter will increase when:

- The acceleration value of an external axis (and the robot) is decreased (*Acc Set*, first parameter).
- The acceleration derivative is decreased (*Acc Set*, second parameter).
- The speed is increased.
- The distances between closely programmed positions are decreased.
- The number of simultaneously controlled axes is increased.
- Using coordinated interpolation.
- Using Weldguide.
- Using Conveyor tracking.
- Using RAPID controlled path correction.
- Using Multitasking with computationally demanding RAPID programs.
- Reorienting with a small or no TCP movement.

It is important to use a path resolution value which is as small as possible in order to achieve a high path resolution also at high speed. Keeping **Path resolution** small can also give shorter cycle times if the cycle contains many stop points, and the move instructions following these stop points, have low speed.

• Choose **Topics: Manipulator**.

• Choose **Types: motion system**.

• Press Enter [↵].

• Select **Path resolution** and change its value.

• Press ***OK*** to confirm.

There is also a RAPID instruction called **PathResol** which affects the resolution of the path. For more information about the instruction see RAPID Reference Manual.

Prefetch time

Prefetch time affects the point in time at which the controller starts to plan for the motion through a fly-by point. If the planning time is too short, a fly-by point will become a stop point. This generates an error called “50024 Corner path failure”. Increasing the parameter **Prefetch time** may solve the problem, when the planning time is too short because of high CPU loading. However, it will not solve the problem when the error occurs because too many fly-by points were placed very closely together, or because of incorrect use of instructions, e.g. a fly-by point followed by a WaitDI instruction. Normally **Prefetch time** should only be increased when the fly-by point is really needed in the application. When it is not really needed, change the fly-by point to a fine point.

There is a drawback when increasing the **Prefetch time**. The difference between the position of the executed RAPID instruction and the current position of the manipulator will increase. This means that after pressing stop during program execution, the program counter on the teach pendant may show an instruction that has not yet affected the manipulator. When starting again, the manipulator will continue along the original path.

• Choose **Topics: Manipulator**.

• Choose **Types: motion system**.

• Press Enter ↵.

• Select **Prefetch time** and change its value.

• Press ***OK*** to confirm.

CPU equalisation

The parameter **CPU equalisation** affects the CPU load in terms of peak load versus average load. When there is a CPU load problem, indicated for example by “50082 Deceleration too long”, then one solution could be to use **CPU equalisation** to distribute the CPU load over time in some other way. Sometimes a higher peak load can be acceptable, as long as it occurs at a favourable moment in time. Try changing **CPU equalisation** both upwards and down, to find the optimal value.

• Choose **Topics: Manipulator**.

• Choose **Types: motion system**.

• Press Enter ↵.

• Select **CPU equalisation** and change its value.

• Press ***OK*** to confirm.

When **CPU equalisation** is increased, it can also be beneficial to increase the system parameter **Queue time**.

Queue time

Increasing **Queue time** will make the system more tolerant to uneven CPU loads. The

drawback is that the robot will react more slowly when jogging and when stopping a program in execution. However, the emergency brake is not affected. Also the accuracy of sensor process, e.g. Weldguide and Conveyor tracking, may be impaired.

• Choose **Topics: Manipulator**.

• Choose **Types: motion system**.

• Press Enter ↵ .

• Select **Queue time** and change its value.

• Press ***OK*** to confirm.

Note that the real queue time is a multiple of a sample time related to dynamic resolution. If the value of the parameter **Queue time** is not an even multiple of dynamic resolution, then the controller will automatically use a queue time as close as possible to the given **Queue time**.

Process update time

Process update time determines how often the process path information is calculated. This information is used for path following in Conveyor, Weldguide and Rapid Weave, for example. Decreasing **Process update time** will improve accuracy but also increase CPU loading. Increasing **Process update time** will decrease the CPU loading.

Note When running programs where the manipulator is moving at high speed, **Process update time** should be kept small to get the best performance. When the manipulator is moving slowly, **Process update time** is not critical.

Choose **Topics: Manipulator**.

• Choose **Types: motion system**.

• Press Enter ↵ .

• Select **Process update time** and change its value.

• Press ***OK*** to confirm.

7.24 Installation optimization of drive system parameters

When installing the software, all the drive system related parameters are set to their nominal values. If the following parameters are adjusted according to the installation (cable length, peak power, mains tolerance) cycle times can be shorter.

Changes outside the range of the installation may violate robot performance.

Mains tolerance

The mains tolerance is set to +10% / -15% on delivery. If the minimum tolerance is less than 15%, cycle times can be shorter if the parameter is changed.

Choose **Topics: Manipulator**.

• Choose **Types: Mains**

• Press Enter ↵ .

• Select **Mains tolerance min** and change its value.

• Press ***OK*** to confirm.

Maximum transformer power

The software assumes a predefined maximum power consumption that together with the mains tolerance gives a certain DC link voltage. If the application needs less than this power, the DC voltage will always be higher and the cycle times can be shorter, if the parameter is changed.

Choose **Topics: Manipulator**.

• Choose **Types: Trafo**

• Press Enter ↵ .

• Select **Power max** and change its value.

Press ***OK*** to confirm.

Cable length

The lengths of the power cables are set to 30 m on delivery. If shorter cables are used this parameter can be changed.

Choose **Topics: Manipulator**.

• Choose **Types: Cable**

• Press Enter ↵ .

• Select **Length** and change its value.

Press ***OK*** to confirm.

CONTENTS

Page

1 Program/Data Storage 3
2 The FileManager Window 4
2.1 Choosing a directory 4
2.2 Viewing file information 4
3 Creating or Moving Files and Directories 5
3.1 Creating a new directory 5
3.2 Renaming a file or a directory 5
3.3 Deleting a file or directory 6
3.4 Copying files and directories 6
3.5 Moving files and Directories 7
3.6 Printing files 7
4 Formatting a Diskette 7

File Manager

The File Manager is used to

- copy or transfer files,
- change the name of a file,
- create directories on diskettes or other mass storage devices,
- print files,
- format diskettes.

1 Program/Data Storage

Programs and data are stored as normal PC text files. These can be saved and restored to/from a diskette or an internal RAM disk.

The *diskette* is a standard 3.5", High Density, 1.44 Mbytes, DOS formatted diskette.

Note. Before saving programs and data, the diskette should be formatted in the robot or in a PC. Pre-formatted DOS diskettes will not always operate satisfactorily.

Note. The diskettes must never be stored inside the cabinet as the information on them can be destroyed due to heat and magnetic fields.

The internal *RAM disk* is a special part of the robot's memory, and can be used in the same way as a diskette.

A *file* can be a program, data created by the program or system parameters and the like, stored in some sort of mass storage.

Directories are used to group files together to achieve a memory unit that is more structured. For example, test programs in one directory and production programs in another (see Figure 1).

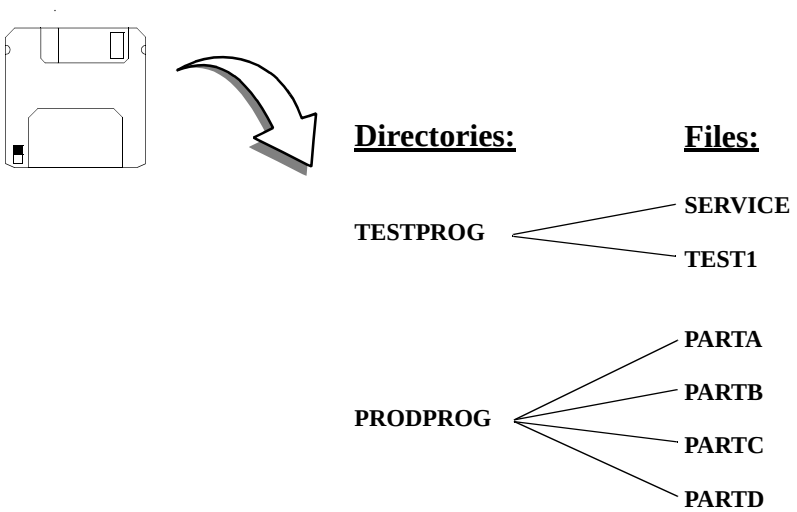


Figure 1 The files can be stored in directories on a diskette.

2 The FileManager Window

- Press the Miscellaneous key .
- Select ***FileManager*** in the dialog box that appears.
- Press Enter ↵ .

The FileManager window will be displayed (see Figure 2)

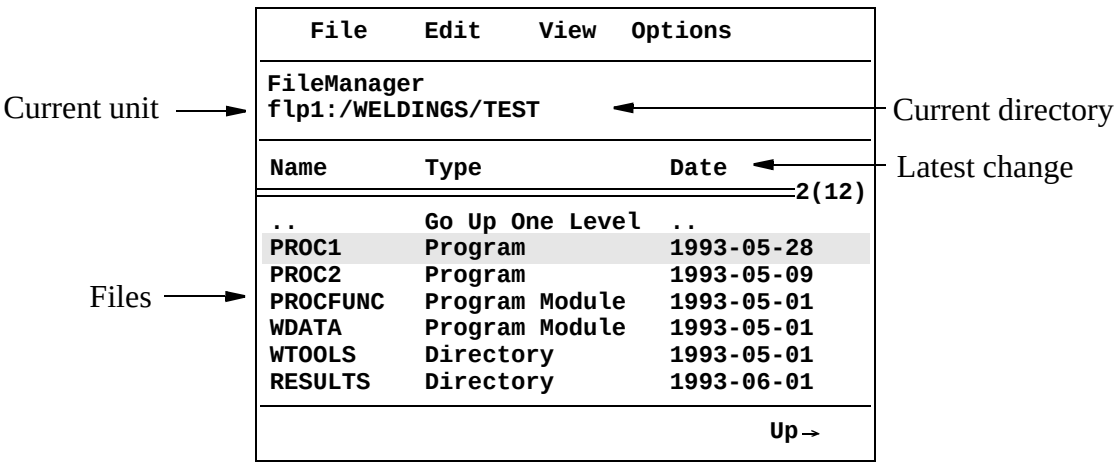


Figure 2 The FileManager window displays all files in a directory.

- Choose the desired unit from the **View** menu:
  - Diskette **View: [flp1:]**
  - RAM disk **View: [ram1disk:]**

2.1 Choosing a directory

- Select the desired directory.
- Press Enter ↵ .

The directories and files located in the chosen directory will be displayed. The next directory above this can be selected by moving to the top line in the list (..) and then pressing Enter ↵ , or by using the ***Up*** function key.

2.2 Viewing file information

- Select a file in the list and press Enter ↵ .

The following information will be specified:

- the name and type of the file,
- the size of the file in bytes,
- the date and time when the file was last changed.

- Choose ***OK*** to terminate the dialog.

3 Creating or Moving Files and Directories

3.1 Creating a new directory

• Choose **File: New Directory.**

A dialog will be displayed, as in Figure 3.

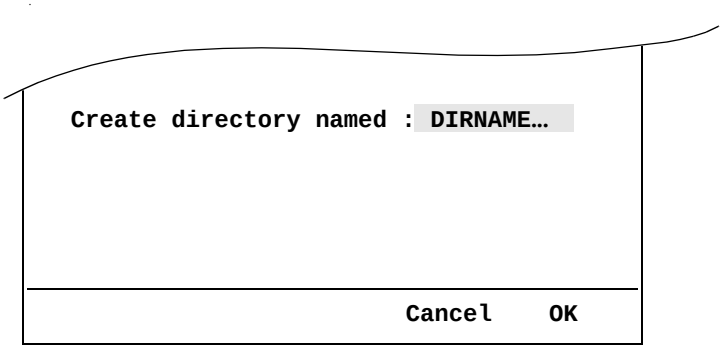


Figure 3 The New Directory dialog.

• Press Enter ↵.

• Enter the new name and press ***OK***.

Confirm by pressing ***OK***. The directory will be created under the current directory.

3.2 Renaming a file or a directory

• Choose **File: Rename**.

A dialog will be displayed, as in Figure 4.

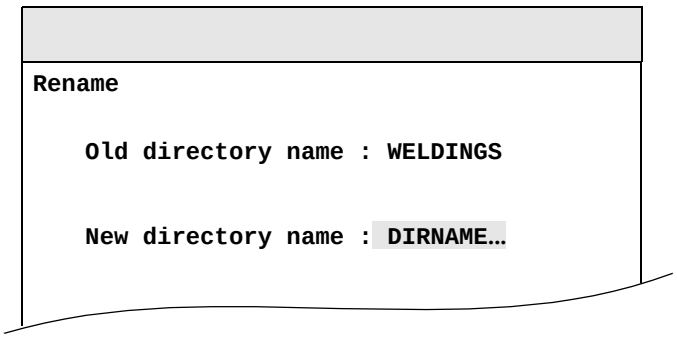


Figure 4 The Rename dialog for a directory.

• Press Enter ↵.

• Enter the new name (max. 8 characters) and press ***OK***.

• Confirm by pressing ***OK***.

3.3 Deleting a file or directory

- Select the desired file or directory.
- Press Delete ⌫ .
- Choose ***OK*** to confirm the deletion.

You can only delete a directory if it is empty.

3.4 Copying files and directories

- Select the file or directory to be copied. If you select a directory, all subordinate directories and files will also be copied.
- Choose **File: Copy.**

A dialog will be displayed, as in Figure 5.

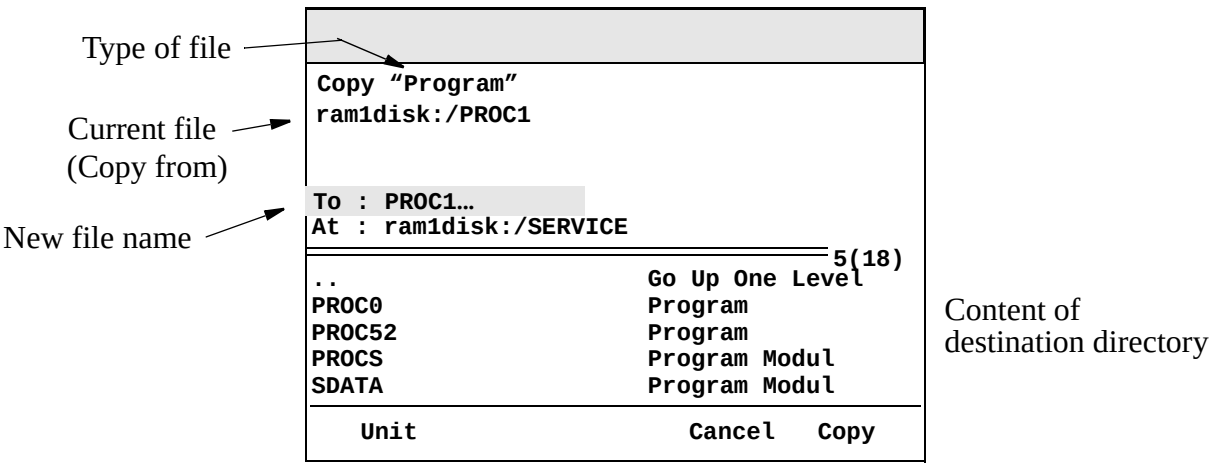


Figure 5 The dialog for copying files or catalogues.

- Specify the name of the new file by selecting the field ***To***, and press Enter ↵ . If you do not specify a name, the copied file/directory will be given the same as the original.
- Specify the destination unit (first part of **At** field) by pressing the Unit function key. If you do not specify a unit, the same unit that was used originally will be used.
- Specify the destination directory (latter part of **At** field) by selecting the lower part of the window. Select the desired directory and press Enter ↵ . If you do not specify a directory, the same directory that was used originally will be used.
- Choose ***Copy*** to start copying.

3.5 Moving files and Directories

- Select the file or directory that is to be moved.
- Choose **File: Move.**

A dialog will be displayed, as in Figure 6.

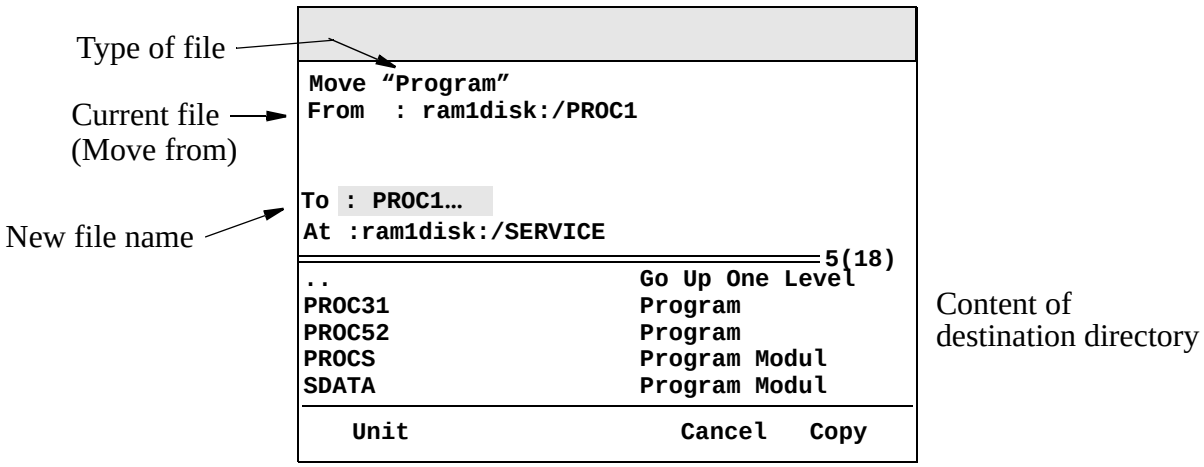


Figure 6 The dialog for moving of files and catalogues.

- Give the file to be moved a new name by selecting ***To***, and press Enter ↵. If you do not specify a new name, the file/directory that is moved will retain the same name.
- Specify the destination unit (first part of **At** field) by pressing the Unit function key. If you do not specify a unit, the same unit that was used originally will be used.
- Specify the destination directory (latter part of **At** field) by selecting the lower part of the window. Select the desired directory and press Enter ↵.
- Choose ***Move*** to start moving.

3.6 Printing files

- Select the file to be printed.
- Choose **File: Print File**
- Choose ***OK*** to start printing.

4 Formatting a Diskette

NB: The previous contents on the diskette will be erased when formatting.

- Choose **Option: Format**.

A confirmation dialog will be displayed.

- If desired, rename the diskette and press Enter ↵.
- Choose ***OK*** to start formatting.

CONTENTS

Page

1 The Service Window 3
2 Changing the Current Date and Time 3
3 Logs 4
3.1 What is a log? 4
3.2 What types of logs are there? 4
3.3 Viewing all logs 5
3.4 Viewing a message in a log 6
3.5 Erasing the contents of a log 6
3.6 Erasing the contents of all logs 6
3.7 Updating the contents of a log automatically or by means of a command 7
3.8 Avoiding normal error reports 7
3.9 Saving log messages on diskette or some other mass storage device 7
4 Calibration 8
4.1 What is calibration? 8
5 Commutation 9
5.1 What is commutation? 9
6 Frame Definition 9
7 Two Axes Definition 9
8 Obtaining information on the robot system 9
9 Backup and Restore 10
9.1 Perform a Backup 10
9.2 Perform a Restore 11
10 Perform a Restart 11

Service

The Service window is used to

- obtain information on the robot system
- view and change logs (e.g. error log)
- calibrate the measuring system for the robot and external axes
- commutate the motors for the robot and external axes
- set the date and time.

For more detailed information on service, maintenance and troubleshooting, see the Product Manual.

1 The Service Window

• Press the Miscellaneous key to open the Service window.

• Select ***Service*** in the dialog box that appears.

• Press Enter .

The service window comprises a number of different windows:

Window title	Used to:
Service Date & Time	Change the current date and time
Service Logs	View logs.
Service Calibrate	Test/Calibrate the measuring system for the robot or external axes.
Service Commutate	Test/Commutate the motors for the robot or external axes.
Frame Definition	Define base or user coordinate system.
Two Axes Definition	Define user frame for a two axes rotational mechanical unit
System Info	Obtain information about storage capacity, task states, system and product ID etc.

2 Changing the Current Date and Time

• Choose **View: Date & Time**.

A dialog box will be called up displaying the current date and time (see Figure 1).

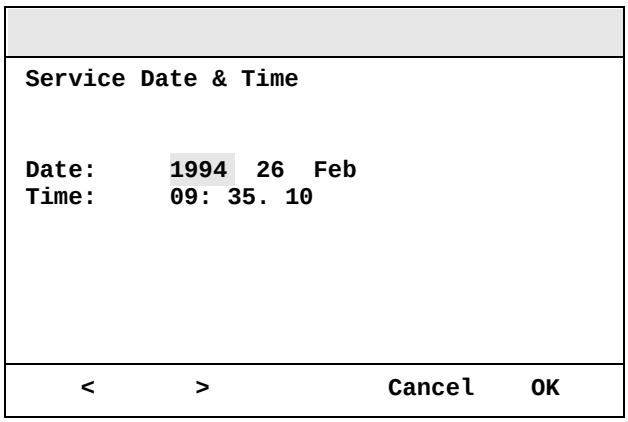


Figure 1 The dialog box used to set the date and time.

- Select that which you wish to change using the arrow keys.
- Using the function keys, < (decreases) and > (increases), change the date or time.
- Choose ***OK*** to confirm.

3 Logs

3.1 What is a log?

All messages reported, such as error messages and changes in the status, are stored in a log. Each message stored is timestamped and it is thus possible to determine the order of events from a log.

When the maximum number of messages in a log is attained, a new message will replace the oldest one.

3.2 What types of logs are there?

The following logs exist:

Name	Max. limit	Used to show
Common	40	All messages
Operational	20	Changes in the status, e.g. a change of operating mode
System	20	The messages related to the control program
Hardware	20	The messages related to defective hardware components
Motion	20	Any messages that appear when moving the robot or other mechanical units
Program	20	Any messages displayed during program execution
Operator	20	Any messages that appear when using the teach pendant
I/O & Communication	20	The messages related to I/O and communication
User	20	User defined messages (by using the instruction ErrWrite)
Arc Welding	20	The messages related to the arc welding process
Spot Welding	20	The messages related to the spot welding process
Internal	20	Internal errors – does not usually contain any messages

3.3 Viewing all logs

• Choose **View: Log**.

The window will display information on all logs in the robot (see Figure 2).

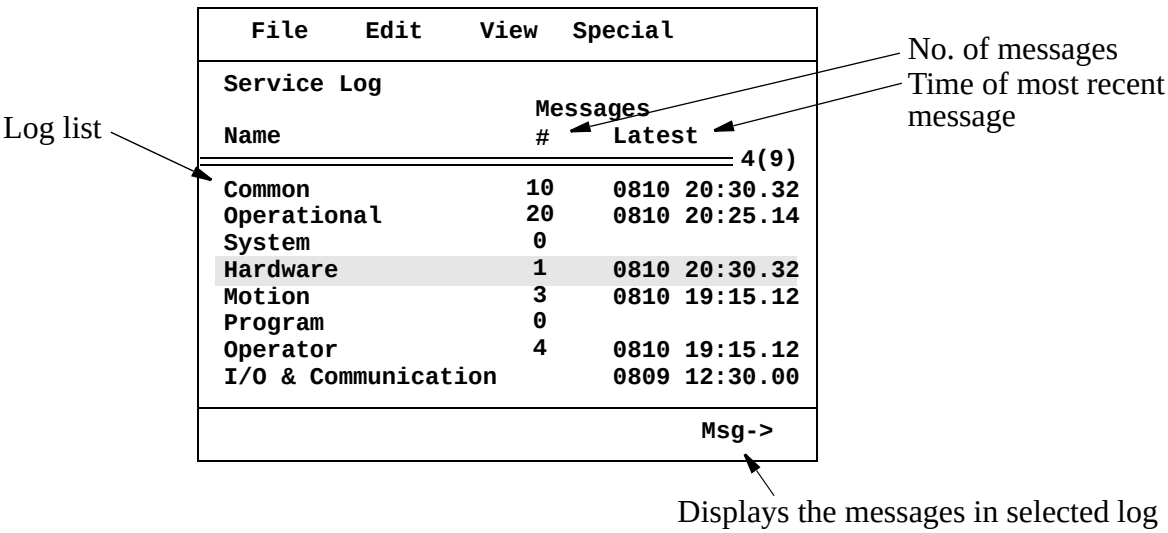


Figure 2 The Service Log window displays all existing logs.

3.4 Viewing a message in a log

- Open the Log window by choosing **View: Log**.
- Choose the log you wish to look at by selecting that log from the list and pressing the ***Msg*** function key, or press Enter ↵ .

The window will display all messages for the log that you choose (see Figure 3).

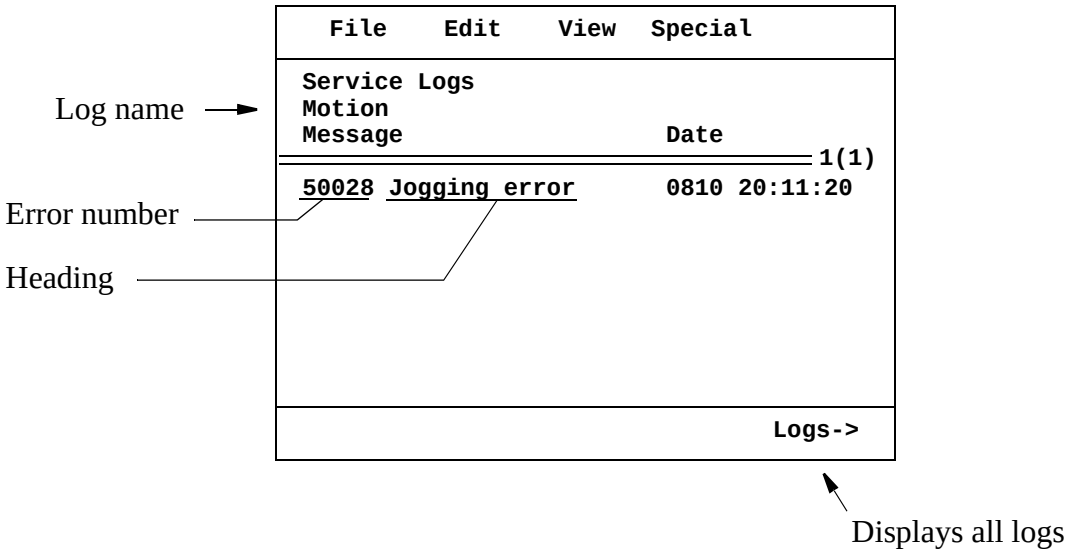


Figure 3 The Service Log Messages window displays all messages in the log.

- You can obtain more information on a specific message by selecting the message and pressing Enter ↵ , or by choosing **Edit: Info**.

3.5 Erasing the contents of a log

- Open the Log window by choosing **View: Log**.
- Select the log to be erased.
- Choose **Special: Erase Log**.
- Choose ***OK*** to confirm.

3.6 Erasing the contents of all logs

- Open the Log window by choosing **View: Log**.
- If there are log messages displayed, press the function key ***logs***.
- Choose **Special: Erase All Logs**.
- Choose ***OK*** to confirm.

3.7 Updating the contents of a log automatically or by means of a command

When you view a log message and a new message appears, you have two choices: you can either update the log

- automatically when the message appears; or
- update the log using the function key ***Update***.
  (The ***Update*** function key is only visible if there are more messages.)

To update automatically:

• Choose **Special: Update log on Event.**

To update on command:

• Choose **Special: Update log on Command.**

3.8 Avoiding normal error reports

When trying to isolate faults in different hardware components, you may not wish to be shown error alert boxes. To prevent these appearing:

• Open the Log window by choosing **View: Log**.

• Choose the Common log by selecting it and pressing the ***Msg*** function key, or press Enter [↵].

Now, error alert boxes will not be shown. Error messages will be either displayed directly (if you chose **Special: Update log on Event**) or following a command in the log (if you chose **Special: Update log on Command**).

3.9 Saving log messages on diskette or some other mass storage device

• Open the Log window by choosing **View: Log**.

• Select the log that you wish to save and choose **File: Save As**.

A dialog box will appear, displaying the contents of the current directory (see Figure 4).

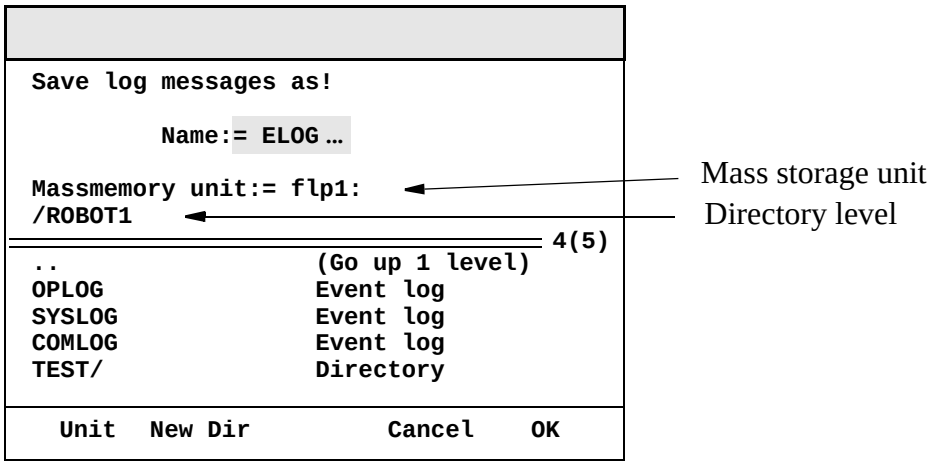


Figure 4 Dialog box for storing logs.

- If necessary, change the mass storage unit by pressing the ***Unit*** function key until the correct unit is displayed. To store on a diskette, choose ***flp1:***.
- Select the ***Name*** field, press Enter ↵ and enter the new name in the dialog box that appears. Choose ***OK*** to confirm.
- Select the directory to which the log is to be saved. You can move to the next directory level by selecting the desired directory or '. .' (upwards) and pressing Enter ↵.

  Create a new directory by pressing the ***New Dir*** function key. Specify the new directory name in the dialog box that appears. Choose ***OK*** to confirm.

- Choose ***OK*** to confirm the save.

4 Calibration

4.1 What is calibration?

Calibration involves setting the calibration positions of the axes and is used as the basis for their positioning. If the robot or external axes are not correctly calibrated, this will result in incorrect positioning and will have a negative effect on the agility of the robot. The robot is calibrated on delivery.

For more information see *Calibrating the robot* in Chapter 10 Calibration in this manual.

5 Commutation

5.1 What is commutation?

Each motor must be commutated in order to be able to utilise it to its full capacity. Commutation involves reading the resolver value when the motor is in a given pose. The robot motors are commutated on delivery.

For information on how to do this, see the section on *Repairs* in the Product Manual.

6 Frame Definition

See *Frames* in Chapter 10 Calibration in this manual.

7 Two Axes Definition

See *Frames* in Chapter 10 Calibration in this manual.

8 Obtaining information on the robot system

• Choose **View: System Info.**

A list of topics is shown in the dialog box (see Figure 5).

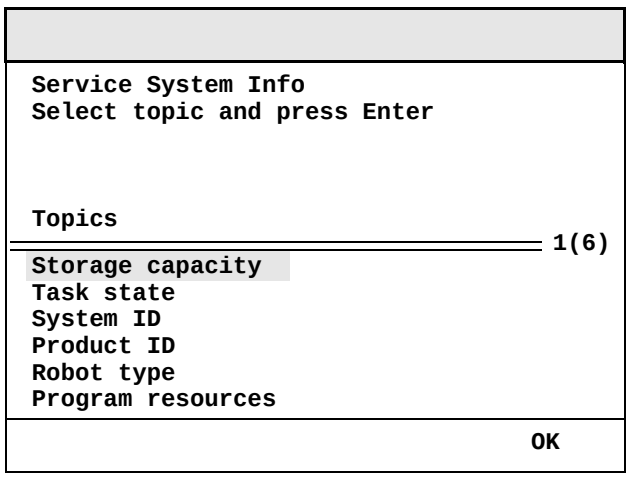


Figure 5 The system information window.

• Select a topic using the arrow keys and press Enter ↵. Information on the selected topic will be displayed.

Topic	Description
Storage capacity	All available storage devices are shown in a list containing the device name, free space and total size.
Task state	All tasks are shown in a list containing task name and task state (Uninitialised, Ready, Executing or Stopped). If a task is *stopped* or *executing* the current instruction is displayed.
System ID	The unique system identification code is shown.
Product ID	The identification code for all installed products is shown.
Robot type	Shows the robot type specification.
Program resources	Shows the total program memory before task configuration and the maximum number of persistents.

• Press ***Update*** to update the information.

9 Backup and Restore

The backup function saves all system parameters, system modules and program modules in a context. If desired, all logs are also saved. The data will be saved in a directory specified by the user.

The restore function retrieves data from a backup directory. Restore replaces all system parameters and loads all modules from the backup directory. A warm start is then automatically performed.

9.1 Perform a Backup

• Choose **File: Backup.**

The Backup dialog will be displayed (see Figure 6).

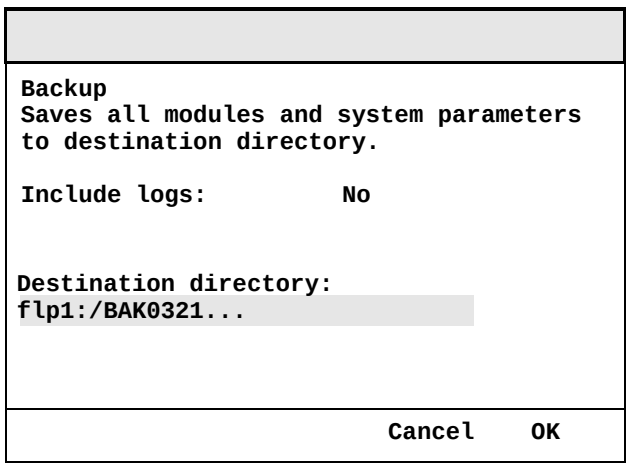


Figure 6 The Backup view.

- Choose whether or not all logs are to be saved by selecting the ***Include logs*** field.
- Select a backup directory. A default directory is suggested, consisting of a prefix and current date, BAKmmdd.
  If you want to change directory press Enter ↵ and select, or create a new, directory.
- Press ***OK*** to start the backup.

9.2 Perform a Restore

- Choose **File: Restore**. The Restore dialog will be displayed.
- Select a source directory generated by the Backup function.
- Press ***OK*** to start the restore.

Warning! Restore will remove and replace all existing parameters and modules in the system.

10 Perform a Restart

In the Service window a number of different restarts can be performed.

- Choose **File: Restart** .
  - Normal restart: Press ***OK***.
  - Generate a cold start to be able to load in a complete new control program: Enter the numbers 1 3 4 6 7 9 and press ***C-Start.***
  - Change language, options or robot type: Enter the numbers 1 4 7 and press ***I-Start.***
  - Activate arc welding parameters: Enter the number 2 5 8 and press ***P-Start***.
    Note When an arc welding parameter is activated, the program memory will be erased. So, make sure that you have saved all programs on diskette.

Fore more information of C-Start and I-Start, see Product Manual - *Installation and Commissioning*.

CONTENTS

Page

1 User screen 3
2 The ScreenViewer Window 3
3 The Screen Options 4
4 The Screen Loading 4
5 The Screen Information 5
6 The Screen Display 6

ScreenViewer

SrceenViewer is used to

- display user screen packages installed
- load a user screen package
- remove an installed user screen package
- list the user screen packages installed
- display information on the user screen packages installed

11 User screen

The *user screen* is a screen composed of text or lines, function keys, pop-up menus and input fields defined by the ScreenMaker PC tool. These user screens can be grouped in a user screen package file under a specific name (Ex: SpotTimer 1.00).

These user screen packages are loaded, removed and displayed via the ScreenViewer window.

For more detailed information on the user screen packages, see the Product Specification RobotWare.

12 The ScreenViewer Window

• Press the Miscellaneous key .

• Select ***Screen Viewer*** in the dialog box that appears.

• Press Enter .

The ScreenViewer window comprises a number of different windows:

Window title	Used to:
Screen Options	select from the three options below:
Screen Loading	load a user screen package.
Screen Information	list the user screen packages installed.
Screen Display	display the user screen packages installed.

13 The Screen Options

The screen options window is always available if no user screen packages have been installed. If at least one user screen package has been installed, the screen options window will be available

- for one minute after the start-up if the controller is in the manual mode,
- never if the controller is in the automatic mode during the start-up.

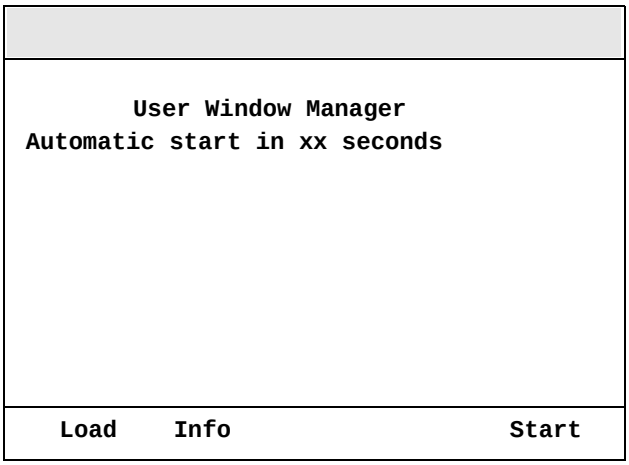


Figure 7 The screen option window.

The choices available are :

- ***Load*** to display the screen loading window.
- ***Info*** to display the screen information window (and also to remove packages)
- ***Start*** to display the screen display window

If the screen option window is not available, the screen display window will be displayed automatically, and the controller will need to be restarted in manual mode to return to the screen options window.

14 The Screen Loading

A user screen package file can be loaded via the screen loading window. The file will be loaded, analysed and installed as a new user screen package.

• Press ***Load*** in the screen options window.

A dialog box appears, displaying all user screen package files (as a parameter file) in the current directory (see Figure 8).

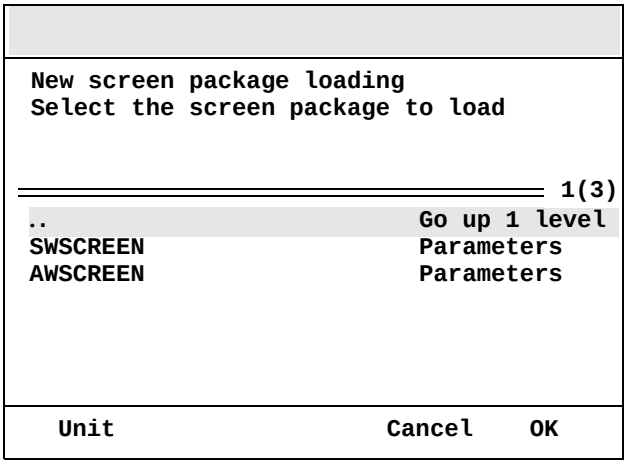


Figure 8 The screen loading window.

- If necessary, change the mass storage unit by pressing the function key ***Unit*** until the correct unit is displayed. To load user screen package files from a diskette, choose ***flp1:***.
- Select the directory from which the user screen package files are to be loaded. You can move to the next directory level by selecting the desired directory or '..' (upwards) and pressing Enter ↵ .
- Choose ***OK*** to confirm the load.

An alert box will be displayed after reading, with the state of the loading. After this the display returns to the screen options window.

15 The Screen Information

- Press ***Info*** in the screen options window:

A dialog box appears, displaying the list of the installed user screen packages (name and size used), and the total memory size used (see Figure 9).

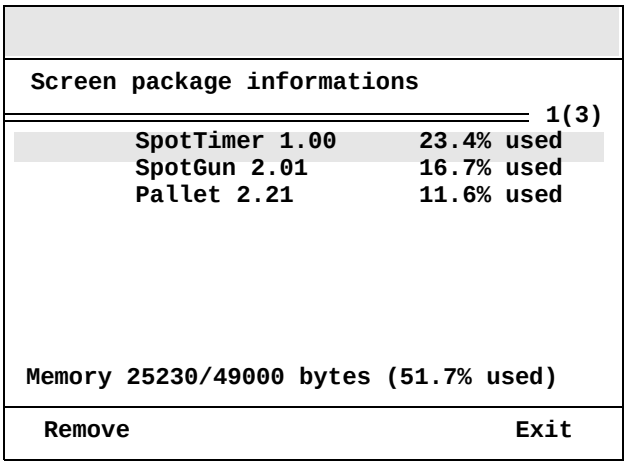


Figure 9 The screen information window.

• Press ***Remove*** to remove the selected package.

• Choose ***Exit*** to return to the screen options window.

16 The Screen Display

The screen display window is accessible only if at least one user screen package has been installed.

• Press ***Start*** in the screen options window:

A dialog box appears, displaying the current screen of the first user screen package installed. Four of the menu keys can be user-defined and represent the definition of the user screen. The menu key **View** is reserved for displaying the list of the installed packages (see Figure 10).

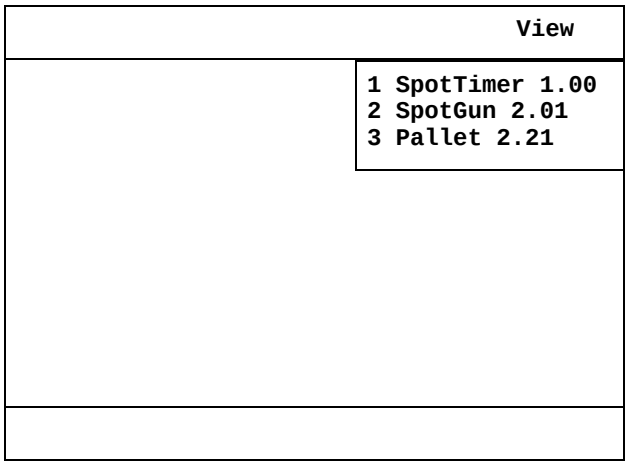


Figure 10 The screen display window.

The selection of the different screens of a user screen package to display is made by the menus and the function keys according to the definition of the user screen package installed.

The selection of a new user screen package is made by the selection of the package in the **View** menu. The current screen of the package selected will then be displayed.

For more detailed informations on a specific user screen package installed, see the documentation given with this user screen package.

CONTENTS

Page

1 Error Management 3
1.1 Confirming an error message 3
1.2 Calling up suggestions on how to correct an error 3
1.3 Acknowledging warning messages 4

Error Management

1 Error Management

If an error occurs, an error message will be displayed in plain language on the teach pendant (see Figure 1). If several errors occur simultaneously, the error that occurred first will be selected.

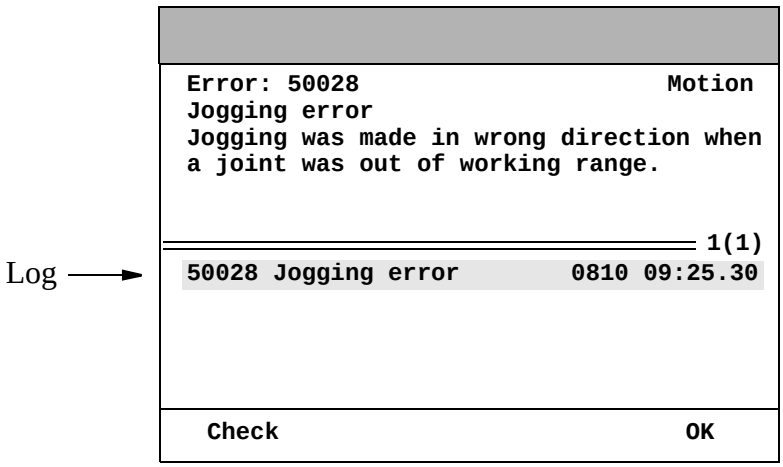


Figure 1 An error message is displayed in plain language as soon as an error occurs.

All errors and status changes are also registered and time-stamped in a log. For more detailed information on these logs, see Service in Chapter 14 of this manual.

1.1 Confirming an error message

• Press ***OK***.

The window displayed before the error occurred will be displayed once more. If you want to view an error message later on, you can find it in the log (see Service in Chapter 14 of this manual).

1.2 Calling up suggestions on how to correct an error

• Press ***Check***.

Information about possible corrective measures is displayed, along with the reason for the error (see Figure 2).

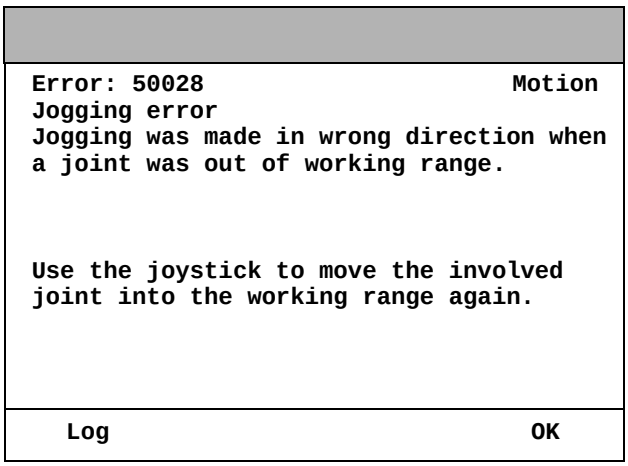

Error: 50028 Motion
Jogging error
Jogging was made in wrong direction when a joint was out of working range.
Use the joystick to move the involved joint into the working range again.
Log
OK

Figure 2 Suggestions on how to correct an error.

• Press ***Log*** to display the log instead of the check list.

1.3 Acknowledging warning messages

Sometimes, a warning or information message will be displayed. This message is displayed in the form of a minimised alert box that conceals only part of the previous window.

• Acknowledge the message by pressing Enter ↵.

CONTENTS

Page

1 Operational error messages 7
2 System error messages 9
3 Hardware error messages 16
4 Program error messages 31
5 Motion error messages 58
6 Operator error messages 68
7 IO & Communication error messages 72
8 Arcweld error messages 83
9 Spotweld error messages 92
10 Paint error messages. 93

1 Operational error messages

10002: Program reset
The task %!%!%s has
been rewound to its start point.

10005: Program stopped
The task %!%!%s has
stopped. The reason is that
%s

10007: Program started
The task %!%!%s has
start to execute.
%s

10008: Program restarted
The task %!%!%s has
restart to execute.
%s

10009: Work memory full
No memory left for new RAPID
instructions or data.
The task is %!%!%s
Check:
Save the program and then
restart.

10010: Motors off state

10011: Motors on state

10012: Guard stop state
Runchain opened by any safety guard
except the emergency stop.
Check:
Close runchain, it could be broken
by the following devices (if used):
Access gate, light screen, servo
disconnector or any other safety
device connected to the run chain.

10013: Emergency stop state
Runchain opened by emergency stop.
Em stop reset is required.
Use the motors off button.

10014: System failure state
Fatal non recoverable system error.
Warm start is required.

10015: Manual mode selected

10016: Automatic mode requested

10017: Automatic mode confirmed

10018: Manual mode FS requested
Manual mode with full speed requested
Check:

10019: Manual mode FS confirmed
Manual mode with full speed confirmed

10020: Execution error state
The program execution has reached
a spontaneous error state

10021: Execution error reset
The program execution has left
a spontaneous error state

10022: Hold to run waiting
Waiting for hold to run button to be
pressed on the programming unit.

10023: Hold to run timeout
The hold to run button on the
programming unit must be pressed
within timeout limit.

10024: Collision triggered
A collision has been detected.

10025: Collision resetted
The collision detection has
been resetted.

10026: Collision confirmed
Returned to the path after
a collision detection.

10027: Collision not confirmed
The system has not returned
to the path after a collision
detection.

10030: All axes commutated

10031: All axes calibrated

10032: All rev counters updated

10033: All axes synchronized

10034: Axis not commutated

10035: Axis not calibrated

10036: Rev counter not updated

10037: Axis not synchronized

10040: Program loaded
The task %!%!%s has
loaded a program or program module.
The free user space was %i bytes
before this operation and
%i bytes after.

10041: Program erased
The task %!%!%s has
erased a program.

10043: Restart failed
The task %!%!%s
can’t restart

10044: Program Pointer updated
The task %!%!%s
could have changed the PP pos.

10045: System restarted
An already installed system
was restarted.

10046: System restarted in cold
mode
First start after installation.

10047: Background task %s
refuse to start
%s

10048: Background task did stop
The task %s stoped without reason
%s

10049: Protected area not finish
A power fail did occur in the
middle of a protected area for
the task %!%s
%s

10050: Execution cancelled
The restart will clear the execution
in task %!%.16s of a
%s

10051: Event routine error
The task %!%s could not start
the specified system event routine
%s
The routine is either unknown to
the system or unlinkable.
Check:
Insert the routine in a system
module or correct the program.

10052: Regain start
A regain movement has started

10053: Regain ready
The regain movement is ready

10060: Test of enable chain
The enable chain is always tested at
startup. If the test failed an error
message concerning enable will follow.
Check:
If enable chain test at startup failed
the related error message will be
"Enable chain timeout"

10070: Backup step ready
The backup %!%s is ready

10071: Backup error
Error during the backup
of %!%!%s
%s
Check:
%s

10072: Restore step ready
The restore %!%s
is ready

10073: Restore error
Error during the restore
of %!%!%s
%s
Check:
%s

10074: NFS server up
The connection to the NFS server
'%s' is working.
All devices remotely mounted from
this server are now available.

10075: NFS server down
The connection to the NFS server
'%s' has been lost.
All devices remotely mounted from
this server are unavailable.

10080: Background task %s
has an older version of a module
installed than the source
%s
Check:
Restart the system with a P-START
to install the newer version

10081: Background task %s
failed to load a newer version of a
module The source of the module is
%s
Check:
See previous messages for the cause
Or restart the system with a P-START
to load the newer version

10082: RAPID Task supervision
Task %s is not running
the system will be set in SysFail
state. It's now impossible to change
to motors on
%s
Check:
See previous messages for the cause
Restart the system to reset the
error state

10083: RAPID Task supervision
Task %s is not running
the system will be set in motors off
state.
%s
Check:
See previous messages for the cause

10084: RAPID Task supervision
Task %s is not running
the main task will also stop
%s
Check:
See previous messages for the cause

2 System error messages

20010: Em stop state active
Em stop reset is required.
Press the panel button.

20011: Em stop state active
Em stop reset is required.
First release the Em stop button
and then press the panel button.

20012: Sys failure state active
Fatal non recoverable system error.
Warm start is required.
Check:
Switch the mains switch off and on
again if the soft restart command is
ignored or not possible to reach.

20024: Enable chain timeout
Two channel status timeout.
Check:
The acknowledgement for a two channel
enable chain status change was not
received within the expected time.

20030: Axis not commutated
One or several internal drive unit
axes are not commutated.

20031: Axis not calibrated
One or several absolute/relative
measurement axes are not calibrated.

20032: Rev counter not updated
One or several absolute measurement axes are not synchronized.
Check:
Move the robot to the sync position and update the revolution counters.

20033: Axis not synchronized
One or several relative measurement axes are not synchronized.
Check:
Order Motors On and synchronize all mechanical units in the list.

20050: Not allowed command
Not allowed in this operating mode.

20051: Not allowed command
Not allowed when client not in control of the resource (program/motion).

20052: Not allowed command
Not allowed in this cabinet state.

20053: Not allowed command
Not allowed in this manipulator state.

20054: Not allowed command
Not allowed when program is executing.

20060: Not allowed command
Not allowed in Auto mode.

20061: Not allowed command
Not allowed when changing to Auto mode.

20062: Not allowed command
Not allowed in Manual mode.

20063: Not allowed command
Not allowed in Manual full speed mode.

20064: Not allowed command
Not allowed when changing to Manual full speed mode.

20070: Not allowed command
Not allowed in Motors On state.

20071: Not allowed command
Not allowed while changing to Motors On state.

20072: Not allowed command
Not allowed in Motors Off state.

20073: Not allowed command
Not allowed while changing to Motors Off state.

20074: Not allowed command
Not allowed in Guard Stop state.

20075: Not allowed command
Not allowed in Emergency Stop state.
Check:
Em stop reset is required.
Press the panel button.

20076: Not allowed command
Not allowed in System Failure state.
Check:
Fatal non recoverable system error.
Warm start is required.
Switch the mains switch off and on again if the soft restart command is ignored or not possible to reach.

20080: Not allowed command
Not allowed when axis is not commutated.

20081: Not allowed command
Not allowed when axis is not calibrated.

20082: Not allowed command
Not allowed when axis rev counter is not updated.

20083: Not allowed command
Not allowed when axis is not synchronized.

20092: Not allowed command
Not allowed in state
System IO Start Blocked.

20100: Teachpendant in ctrl
A teachpendant application is in control of the requested resource (program/motion)

20101: Teachp (prg) in ctrl
The teachpendant programming window has focus and is in control of the program server.
Change to the production window and perform the command again.

20102: Teachp (joystick) in ctrl
The teachpendant joystick is in control of the motion server.
Release the joystick and perform the command again.

20111: Teachp (prg) in ctrl
The teachpendant programming window has focus and is in control of the program server.
Change to the production window and perform the command again.

20112: Program 1 in ctrl
The program server 1 is in control of the motion server.
Stop the program and perform the command again.

20113: Program 2 in ctrl
The program server 2 is in control of the motion server.
Stop the program and perform the command again.

20114: Program 3 in ctrl
The program server 3 is in control of the motion server.
Stop the program and perform the command again.

20115: Program 4 in ctrl
The program server 4 is in control of the motion server.
Stop the program and perform the command again.

20116: Program 5 in ctrl
The program server 5 is in control of the motion server.
Stop the program and perform the command again.

20120: System IO in ctrl

20125: Client %s in ctrl
Specified client is in control of the requested resource (program/motion)

20130: Out of memory in cfg

20131: Unable to read file

20132: Parameters not saved
Parameters cannot be saved.
Probably, because disk is write protected or no space available.
Check:
Check if disk is write-protected or if space on disk is enough.

20133: Cannot modify instance
DescriptionReason:
Can’t replace instance in line %d of file %s
Check:
The instance is write protected.

20134: Wrong version
DescriptionReason:
The cfg domain version is wrong in file %s
The software is made for version %s
Check:
Change the version of the cfg domain.

20135: Line too long
Description\Reason:
Line %d > %d characters
Check:
Reduce the number of characters.

20136: Attr out of range
DescriptionReason:
Attribute %s is out of range in line %d
Check:
Change the value on the attribute.

20137: Dublicate inst name
DescriptionReason:
Dublicate name in line %d
of file %s
Check:
Change the name.

20140: Motors On rejected.
Motors On via System IO
not allowed.

20141: Motors Off rejected.
Motors Off via System IO
not allowed.

20142: Start rejected.
Start/restart of program via System
IO not allowed.
Check:
The reason could be that the robot
is outside of regain distance.

20143: Start main rejected.
Start of main program via System IO
not allowed.

20144: Stop rejected.
Stop of program via System IO
not allowed.

20145: Stop cycle rejected.
Stop of program cycle via System IO
not allowed.

20146: Man interrupt rejected.
Manual interrupt of program via
System IO not allowed.

20147: Load and start rejected.
Load and start of program via
System IO not allowed.
Program file name (including mass
memory unit) to be loaded must
be defined.

20148: Confirm rejected.
Emergency Stop Reset Confirm via
System IO not allowed.

20149: Error reset rejected.
Program execution error reset via
System IO not allowed.

20150: Syncronization rejected.
Syncronization of mechanical unit
via System IO not allowed.

20151: Faulty signal name.
Signal name not possible to
subscribe to for Sysio.
The Signal name might not be in
the cfg-file for Sysio.

20152: Too many restrictions.
For an action (signal) in
Sysio, no restrictions are set.
The total number of restrictions
(signals) for an action in the
cfg-file for Sysio are too high.

20153: Mot. On, Start rejected.
Motors On, Start/restart of program
via System IO not allowed.
Check:
The reason could be that the robot
is outside of regain distance.

20154: Stop instr. rejected.
Stop of program instruction via
System IO not allowed.

20155: Undefined Argument
SyncExtAx mechanical_unit_name
is not defined

20156: Undefined Argument
Interrupt routine_name
is not defined

20157: Undefined Argument
LoadStart program_name
is not defined

20158: No System Input signal
A system input has been declared to
a signal that doesnt exist.

20159: No System Output signal
A system output has been declared to
a signal that doesnt exist.

20160: Not in configuration
The system module %!%s in task %s has no corresponding specification in the configuration for "Task modules"
Check:
View "Task modules" in the "System Parameter" menu and add an item for this system module

20161: Path not find
The system module %!%s in task %s has a corresponding specification in the configuration for "Task modules" that point out a non existing file path
Check:
View "Task modules" in the "System Parameter" menu and change the path in the item for this system module

20162: Write error
A write error occur when the system try to save the system module %!%.14s
at %.37s
in task %.16s. Or the file system was full
Check:
View "Task modules" in the "System Parameter" menu and change the path in the item for this system module

20163: Reconfig failed
Some user module(s) changed but not saved. See previous warnings stored in the log.
Check:
Save those modules that are specified by earlier warnings and try another system start.

20164: Reconfig failed
There are still some unsaved system module
Check:
Read error descriptions in earlier messages.
Try another system start

20165: PP lost!
Restart is no longer possible from current position. The program has to be started from the beginning.

20166: Refuse to save module
The module %!%.14s
is older than the source
at %.37s
in task %.16s.

20167: Unsaved module
The module %!%.14s
is changed but not saved
in task %.16s.

20170: SYS_STOP
Program motion stopped along path.

20171: SYS_HALT
Program and motion stopped with motors off.

20172: SYS_FAIL
System Failure, restart system.

20175: Teachpendant fail
Teachpendant lost contact
this will force Guard Stop

20180: System IO restriction
System IO restrictions are active for other clients, e.g. teach pendant or computer link.
Check:
Motor on is restricted by System IO MotorOff.
Program start is restricted by System IO Stop, StopCycle or StopInstr.

20201: Limit Switch open

20202: Emergency Stop open

20203: Enabling Device open

20204: Operation Key open

20205: Auto Stop open

20206: General Stop open

20207: Backplane Enable open
Check:
1. Check other error messages for primary fault reason.
2. If no other error messages, please check line voltage for one phase missing.

20208: Chain switches open
One or many switches in chain open.

20209: External Contactor open

20210: Motor Contactor open

20211: Two channel fault
Enable from backplane was not allowed to be closed.
Runchain two channel fault has not been reset.
Check:
First open both channels and then close them again to reset runchain.
Please check the safety guard that caused the status conflict.

20212: Two channel fault
Runchain was not allowed to be closed.
Runchain two channel fault has not been reset.
Check:
First open both channels and then close them again to reset runchain.
Please check the safety guard that caused the status conflict.

20213: Two channel fault
Runchain two channel fault has not been reset.
Check:
First open both channels and then close them again to reset runchain.
Please check the safety guard that caused the status conflict.

20221: Chain conflict
Status conflict
for one or many switches in chain.
Check:
Please check the two channel safety guard that caused the status conflict.

20222: Limit Switch conflict
Status conflict
for the Limit Switch chain.
Check:
Please check the two channel safety guard that caused the status conflict.

20223: Emergency Stop conflict
Status conflict
for the Emergency Stop chain.
Check:
Please check the two channel safety guard that caused the status conflict.

20224: Enabling Device conflict
Status conflict
for the Enabling Device chain.
Check:
Please check the two channel safety guard that caused the status conflict.

20225: Auto Stop conflict
Status conflict
for the Auto Stop chain.
Check:
Please check the two channel safety guard that caused the status conflict.

20226: General Stop conflict
Status conflict
for the General Stop chain.
Check:
Please check the two channel safety guard that caused the status conflict.

20227: Motor Contactor conflict
Status conflict
for the Motor Contactor chain.
Check:
Please check the two channel safety guard that caused the status conflict.

20228: Ordered ES conflict
Status conflict
between ordered and configured type of Emergency Stop Turn Off (immediate or delayed).
Check:
Replace Panel Board.

20229: Ordered AS conflict
Status conflict
between ordered and configured type of Auto Stop Turn Off (immediate or delayed).
Check:
Replace Panel Board.

20230: Ordered GS conflict
Status conflict
between ordered and configured type of General Stop Turn Off (immediate or delayed).
Check:
Replace Panel Board.

20231: Delayed ES conflict
Status conflict
between ES1 and CH1 or ES2 and CH2 after a delayed Emergency Stop.
Check:
Replace Panel Board.

20232: Delayed AS conflict
Status conflict
between AS1 and CH1 or AS2 and CH2 after a delayed Auto Stop.
Check:
Replace Panel Board.

20233: Delayed GS conflict
Status conflict
between GS1 and CH1 or GS2 and CH2 after a delayed General Stop.
Check:
Replace Panel Board.

20234: Immediate ES conflict
Status conflict
between ES1 and CH1 or ES2 and CH2 after an immediate Emergency Stop.
Check:
Replace Panel Board.

20235: Immediate AS conflict
Status conflict
between AS1 and CH1 or AS2 and CH2 after an immediate Auto Stop.
Check:
Replace Panel Board.

20236: Immediate GS conflict
Status conflict
between GS1 and CH1 or GS2 and CH2 after an immediate General Stop.
Check:
Replace Panel Board.

20241: Key speed status fault
Status conflict
for the operating mode key signals and the speed signal.
Check:
Check operating mode key hardware or replace Panel Board.

20242: Auto mode conflict
Status conflict
for the operating mode key signals in Auto operation.
Check:
Check operating mode key hardware or replace Panel Board.

20243: Manual mode conflict
Status conflict
for the operating mode key signals in Manual operation.
Check:
Check operating mode key hardware or replace Panel Board.

20244: Manual FS mode conflict
Status conflict
for the operating mode key signals
in Manual Full Speed operation.
Check:
Check operating mode key hardware
or replace Panel Board.

20251: Transformer temp. high
Status active for over temperatur
in main transformer.
Make sure to let the transformer cool
down before ordering Motors On again.
Check:
View Safety in the IO window and wait
until the signal TRFOTMP equals 0
before ordering Motors On again.

20252: Motor temp. high
Status active
for over temperature in
motors of manipulator.
Make sure to let the Motors cool down
before ordering Motors On again.
Check:
View Safety in the IO window and wait
until the signal PTC equals 0 before
ordering Motors On again.

20253: Ext. device temp. high
Status active
for over temperature in
external device.
Make sure to let the Motors cool down
before ordering Motors On again.
Check:
View Safety in the IO window and wait
until the signal PTCEXT equals 0 before
ordering Motors On again.

20254: Power supply fan stopped
Status active
when fan in power supply not running.
Check:
Check the power supply fan hardware.

20255: Panel Board voltage low
Status active
when Panel Board 24 V failed.
Check:
Check the Panel Board voltage.

20260: Run control status fault
Status conflict between
motor contactors and run control.
Check:
Replace Panel Board.

20261: Strings to long
Description\Reason:
- Elog message number %d:
  Total String length %d >
  %d characters
Check:
1. Reduce the total string length.

20270: Access error
Panel Module access error.
Check:
Examine your EIO configuration files.

20280: Symbol conflict
The signal %s defined in the IO
configuration conflict with another
program symbol with the same name.
Due on that fact the signal will not be
mapped to a program variable.
Check:
Rename the signal in
the IO configuration.

3 Hardware error messages

31108: Error in serial channe
Error in serial channel %.f
Check:
1. Check communication parameters
2. Replace robot computer board

31114: Bus error
Bus error when accessing LED on
main computer
Check:
1. Replace main computer board
2. Replace robot computer board

31115: Error in serial channel 1
Received data not equal to transmitted data
Check:
1. Check communication parameters
2. Replace robot computer board

31117: Parity error channel 1
Check:
1. Check communication parameters
2. Replace robot computer board

31118: Framing error channel 1
Check:
1. Check communication parameters
2. Replace robot computer board

31119: Noise error channel 1
Check:
1. Check communication parameters
2. Replace robot computer board

31130: Port error
Check:
Replace robot computer board

31131: SYSRESET did not fire.
Replace VME bus boards.
1. Check Robot computer.
2. Check other VME-bus boards.
3. Check backplane.

31132: Error in serial channel 2
Received data not equal to transmitted data
Check:
1. Check communication parameters
2. Replace robot computer board

31133: Overflow serial channel 2
Check:
1. Check communication parameters
2. Replace robot computer board

31134: Parity error channel 2
Check:
1. Check communication parameters
2. Replace robot computer board

31135: Framing error channel 2
Check:
1. Check communication parameters
2. Replace robot computer board

31136: Noise error channel 2
Check:
1. Check communication parameters
2. Replace robot computer board

31137: Error in serial console
Received data not equal to transmitted data
Check:
1. Check communication parameters
2. Replace robot computer board

31138: Overflow serial console
Check:
1. Check communication parameters
2. Replace robot computer board

31139: Parity error console
Check:
1. Check communication parameters
2. Replace robot computer board

31140: Framing error console
Check:
1. Check communication parameters
2. Replace robot computer board

31141: Noise error console
Check:
1. Check communication parameters
2. Replace robot computer board

31142: Error in tpu channel
Received data not equal to transmitted data
Check:
1. Check communication parameters
2. Replace robot computer board

31143: Overflow in tpu channel
Check:
1. Check communication parameters
2. Replace robot computer board

31144: Parity error tpu channel
Check:
1. Check communication parameters
2. Replace robot computer board

31145: Framing error tpu channel
Check:
1. Check communication parameters
2. Replace robot computer board

31146: Noise error tpu channel
Check:
1. Check communication parameters
2. Replace robot computer board

31203: Floppy Disk Error
Bad floppy disk or not formatted
Check:
1. Repeat attempt
2. Change disk

31206: Floppy Disk Error
Bad floppy disk or internal error
Check:
1. Check the floppy drive
2. Change disk
3. Restart the system

31207: Floppy Disk Error
No floppy disk or disk not ready
Check:
1. Repeat attempt

31210: Floppy Disk Error
Invalid format
Check:
1. Change disk

31211: Floppy Disk Error
Data transfer error to/from floppy
Check:
1. Repeat attempt
2. Change disk
3. Restart the system

31214: Floppy Disk Error
Data transfer was interrupted
Check:
1. Repeat attempt
2. Restart the system

31215: Floppy Disk Error
Internal command invalid
Check:
1. Repeat attempt
2. Restart the system

31216: Floppy Disk Error
Floppy disk was moved during tranfer
Check:
1. Repeat attempt
2. Restart the system

31217: Floppy Disk Error
Bad floppy disk or floppy device
Check:
1. Repeat attempt
2. Change Disk
3. Restart the system

31219: Floppy Disk Error
Floppy device not ready
Check:
1. Repeat attempt
2. Restart the system

31220: Floppy Disk Error
Bad floppy disk or internal error
Check:
1. Repeat attempt
2. Change Disk
3. Restart the system

31221: Floppy Disk Error
Data error
Check:
1. Repeat attempt
2. Change Disk
3. Restart the system

31222: Floppy Disk Error
Internal error - Overrun
Check:
1. Repeat attempt
2. Restart the system

31223: Floppy Disk Error
Bad floppy or internal error
Check:
1. Repeat attempt
2. Change Disk
3. Restart the system

31224: Floppy Disk Error
Floppy write protected
Check:
1. Remove write protection

31225: Floppy Disk Error
Bad Floppy - Address mark missing
Check:
1. Change Disk

31226: Floppy Disk Error
Bad data on floppy
Check:
1. Change Disk

31227: Floppy Disk Error
Bad floppy - Missing cylinder
Check:
1. Change Disk

31228: Floppy Disk Error
Bad floppy - Bad cylinder
Check:
1. Change Disk

31229: Floppy Disk Error
Bad floppy - Bad address mark in data
Check:
1. Change Disk

31401: DMA error
DMA transfer error in ROBOT COMPUTER
Check:
Replace robot computer board

31402: DMA error
DMA transfer error in ROBOT COMPUTER
Check:
Replace robot computer board

31403: DMA error
DMA transfer error in ROBOT COMPUTER
Check:
Replace robot computer board

31404: DMA error
DMA transfer error in ROBOT COMPUTER
Check:
Replace robot computer board

31408: Axis computer error
Check:
Replace robot computer board

31409: Robot computer error
Check:
Replace robot computer board

31410: Axis computer error
Check:
Replace robot computer board

31411: Axis computer error
Check:
Replace robot computer board

31414: Main computer error
Check:
1. Replace main computer board
2. Replace robot computer board

31415: Main computer error
Check:
Replace main computer board

31418: DMA transfer error
DMA transfer error in ROBOT COMPUTER
Check:
Replace robot computer board

31419: DMA transfer error
DMA transfer error in ROBOT COMPUTER
Check:
Replace robot computer board

31420: DMA transfer error
DMA transfer error in ROBOT COMPUTER
Check:
Replace robot computer board

31501: Battery voltage too low
Battery voltage too low on
battery 1
Check:
Replace battery 1

31502: Battery voltage too low
Battery voltage too low on
battery 2
Check:
Replace battery 2

31503: Battery voltage too low
Battery voltage too low on both
batterys
Check:
Replace batterys

31505: Battery cirquit error
Check:
Replace robot computer board

31605: Memory error IO-computer
Check:
Replace robot computer board

31606: Memory error IO-computer
Check:
Replace robot computer board

31607: Memory error IO-computer
Check:
Replace robot computer board

32118: MC RESET ERROR.
Main computer running in spite
of RESET command.
Check:
Replace main computer board

32247: Mailbox 1 interrupt error
Mailbox 1 interrupt error on
IO computer
Check:
Replace robot computer board

32248: Mailbox 2 interrupt error
Mailbox 2 interrupt error on
IO computer
Check:
Replace robot computer board

32301: Memory error MAIN COMP.
Check:
Replace main computer board

32302: Memory error MAIN COMP.
Check:
Replace main computer board

32303: Memory error MAIN COMP.
Check:
Replace main computer board

32305: Type error MEMORY EXPANS.
Check:
Replace memory expansion board

33150: Axis Computer Int Error
Axis computer was stopped with
hw interrupt due to interrupt error
Check:
Reload system
Replace robot computer board

33158: Axis Comp Driver Clk fail
Axis computer driver clock failure
Main computer is not responding on
request
Check:
Reload system
Replace main computer board

33159: Manual Mode Speed Warning
Manual mode speed exceeded for
the joint connected to axc channel %.f.
Check:
Check for correct load mass definition
Check controller parameters on external
axes
Check for robot singularity
Replace drive unit

33201: Axis cpu Read Error
Error in reading from axis computer driver. Axis computer driver did not return correct number of bytes.
Check:
Check system configuration
Reload system
Replace robot computer board

33202: Axis cpu Write Error
Error in writing to the axis computer driver. Axis computer driver did not return correct number of bytes.
Check:
Check system configuration
Reload system
Replace robot computer board

33203: Axis cpu ioctl Error
Error in ioctl to the axis computer driver.
Fail to execute ioctl command
Check:
Restart system
Reload system
Replace robot computer board

33210: Feedback Position Error
Driver failed to read feedback position on joint %.f
Check:
Restart system
Replace main computer board

33211: Position Control Underrun
Unable to complete position control in the allowed time
Check:
Reload system
Check noise level on I/O connections

33212: DMA Time out Error
DMA access failed from main computer to
axis computer
Check:
Reload system
Replace main computer board and
axis computer board

33213: DMA Operation Error
DMA Control Operation failed from Main computer to Axis computer
Check:
Reload system
Replace main computer board and
axis computer board

33214: Float number error
Illegal references sent from Main computer to Axis computer for joint %.f
Check:
Reload system
Check system parameters

33220: Axis computer failure
Axis computer has returned an error code indicating hardware failure
Check:
Reload system
Replace robot computer board

33301: Error in axis computer
Check:
Replace robot computer board

33302: Error in axis computer
Axis computer was not able to deactivate VME signal SYSFAIL
Check:
Replace robot computer board

33303: Error in axis computer
Axis computer was not able to activate VME signal SYSFAIL
Check:
Replace robot computer board

33304: Error in axis computer
Check:
Replace robot computer board

33305: Error in axis comp memory
Check:
Replace robot computer board

33308: Error in axis computer
Check:
Replace robot computer board

33309: Error in axis computer
Check:
Replace robot computer board

33310: Error in axis computer
Check:
Replace robot computer board

33311: Axis computer
Current ref. loopback error
Check:
Replace robot computer board

33312: Axis computer error
RUNNING/DRVFLT signal error
Check:
1. Replace robot computer board
2. Check drive system boards

33314: Axis computer error
Check:
Replace robot computer board

33315: Axis computer error
Check:
Replace robot computer board

33316: Axis comp err loopb comm.
Axis computer error loopback
comm. error.
Check:
Replace robot computer board

33320: Axis computer error.
Error reported by Axis computer
Check:
Replace robot computer board

33321: Axis computer error.
Uncorrect VME-vector generated at test
Check:
Replace robot computer board

33401: Over writing mea. system
Over writing of output data to
measurement system %.f
Check:
Reload system
Replace robot computer board
Replace measure board

33402: Over writing drive sys.
Over writing of output data to
drive system %.f
Check:
Reload system
Replace robot computer board
Replace drive unit

33403: Over writing mea. system
Over writing of input data from
measurement system %.f
Check:
Reload system
Replace robot computer board
Replace measure board

33404: Over writing drive sys.
Over writing of input data from
drive system %.f
Check:
Reload system
Replace robot computer board
Replace drive unit

33405: Timeout mea. system
Contact lost with measurement
system %.f. Axis computer stopped
due to transmission timeout.
Check:
Check connections from cabinet to
measurement board(s).
Replace measurement board or
robot computer

33406: Timeout drive sys.
Contact lost with drive system %.f.
Axis computer stopped due to
transmission timeout.
Check:
Check connections from cabinet to
drive unit(s).
Replace drive modeule or
robot computer

33407: Access violation.
Access violation of measurement system
Check:
Reload system
Replace robot computer board

33408: Access violation.
Access violation of drive system
Check:
Reload system
Replace robot computer board

33409: Access violation.
Access violation of R6 calculation unit
Check:
Reload system
Replace robot computer board

33410: Access violation.
More than one status in R6 cleared simultanously
Check:
Reload system
Replace robot computer board

33411: Unknown error interrupt
Unknown error interrupt from the axis computer
Check:
Restart system
Replace robot computer board

33412: Clock error
Axis computer driver clock failure
Main computer is not responding on request
Check:
Reload system
Replace main computer board

33413: Loopback error
Error in measurement system %d when testing seriel link in loopback mode
Check:
Replace robot computer board

33414: Loopback error
Error in drive system %d when testing seriel link in loopback mode
Check:
Replace robot computer board

33415: Drive system error
Error in the drive system %d when testing seriel link in normal mode
Check:
Check seriel link
Check drive unit
Check dc link
Check/Replace robot computer board

34001: Bus error
Unexpected bus error during a VME-test
Check:
Check VME boards

34002: Bus error
Unexpected reply during a VME-test
Check:
Check VME boards

37001: Contactor activate Error
Motor On contactor did not activate/ energize
Check:
1. Restart system
2. Check/replace contactors (M.On/AUX)
3. Replace panel board

37002: DSQC306 not running
Main computer software not downloaded or not running

37003: Main computer error
Check:
Replace main computer board

37004: Main computer error
Check:
Replace main computer board

37005: Main computer error
Check:
Replace main computer board

37006: Main computer error
Check:
Replace main computer board

37007: Main computer error
Check:
Replace main computer board

37008: Main computer error
Check:
Replace main computer board

37009: Main computer error
Check:
Replace main computer board

37010: Main computer error
Check:
Replace main computer board

37011: Main computer error
Check:
Replace main computer board

37012: Main computer error
Check:
Replace main computer board

37013: Main computer error
Check:
Replace main computer board

37014: Main computer error
Check:
Replace main computer board

37015: Main computer error
Check:
Replace main computer board

37016: Main computer error
Check:
Replace main computer board

37017: Main computer error
Check:
Replace main computer board

37018: Main computer error
Check:
Replace main computer board

37019: Main computer error
Check:
Replace main computer board

37020: Main computer error
Check:
Replace main computer board

37021: Main computer error
Check:
Replace main computer board

37022: Main computer error
Check:
Replace main computer board

37023: Main computer error
Check:
Replace main computer board

37024: Main computer error
Check:
Replace main computer board

37025: Main computer error
Check:
Replace main computer board

37026: Main computer error
Check:
Replace main computer board

37027: Main computer error
Check:
Replace main computer board

37028: Main computer error
Check:
Replace main computer board

37029: Main computer error
Check:
Replace main computer board

37030: Main computer error
Check:
Replace main computer board

37031: Main computer error
Check:
Replace main computer board

37032: Main computer error
Check:
Replace main computer board

37033: Main computer error
Check:
Replace main computer board

37034: Main computer error
Check:
Replace main computer board

37035: Main computer error
Check:
Replace main computer board

37036: Main computer error
Check:
Replace main computer board

37037: Main computer error
Check:
Replace main computer board

37038: Main computer error
Check:
Replace main computer board

37039: Main computer error
Check:
Replace main computer board

37040: Main computer error
Check:
Replace main computer board

37041: Main computer error
Check:
Replace main computer board

37042: Main computer error
Check:
Replace main computer board

37043: Main computer error
Check:
Replace main computer board

37044: Main computer error
Check:
Replace main computer board

37045: Main computer error
Check:
Replace main computer board

37046: Main computer error
Check:
Replace main computer board

37047: Main computer error
Check:
Replace main computer board

37048: Main computer error
Check:
Replace main computer board

37049: Contactor activate Error
Contactor for station %s or
Supervisory contactor or auxiliary
contactor did not activate or energize
Check:
Restart system
Check contactor or Supervisory
contactor or auxiliary contactor
Check digital output to contactor
or digital input from contactor
Replace system board

38001: Battery backup lost
Battery backup on serial measurement
board %.f on measurement system %.f
lost since last
power down or restart
Check:
Check battery voltage during power off
after 18 hours recharging in power on
Check battery connection to serial
measurement board
Replace battery

38010: Serial Board not found
Serial measurement board %.f on
measurement system %.f not found
Check:
Check system configuration parameters
Check connections and cables to
serial measurement system
Replace serial measurement board

38012: Serial Offset X Error
Offset error in X signal on serial
measurement board %.f on
measurement system %.f
Check:
Replace serial measurement board

38013: Serial Offset Y Error
Offset error in Y signal on serial measurement board %.f on measurement system %.f
Check:
Replace serial measurement board

38014: Serial Linearity Error
Linearity error in X-Y signal difference on serial measurement board %.f on measurement system %.f
- System may still operate with warning
- System will not function with error
Check:
Replace serial measurement board

38015: Serial Linear X Error
Linearity error in X signal on serial measurement board %.f on measurement system %.f
Check:
Replace serial measurement board

38016: Serial Linear Y Error
Linearity error in Y signal on serial measurement board %.f on measurement system %.f
Check:
Replace serial measurement board

38030: Resolver error
Failure in X or Y resolver signal on joint %s
Sum of squared X and Y exceeds max
Check:
Check resolver and resolver connections.
Replace measurement boards

38031: Resolver error
Failure in X or Y resolver signal on joint %s
Sum of squared X and Y below min
Check:
Check resolver and resolver connections.
Replace measurement boards

38032: Transmission failure
Axis computer detected failure in transmission to/from serial measurement system %d.
Check:
Check connections/cables for serial measurement system. Check shieldings
Check for high electromagnetic disturbances along cable run to robot
Replace measure board or robot computer

38033: Transmission failure
Axis computer detected failure in transmission to/from serial measurement system %d.
Check:
Check connections/cables for serial measurement system. Check shieldings
Check for high electromagnetic disturbances along cable run to robot
Replace measure board or robot computer

38034: Transmission failure
Axis computer detected failure in transmisson to/from serial measurement system %d.
Accumulated errors since warmstart: %d.
%d absent transmission of %d detected.
Check:
Check connections/cables for serial measurement system. Check shieldings
Check for high electromagnetic disturbances along cable run to robot
Replace measure board or robot computer

38035: Transmission failure
Axis computer detected failure in transmission to/from serial measurement system %d.
Check:
Check connections/cables for serial measurement system. Check shieldings
Check for high electromagnetic disturbances along cable run to robot
Replace measure board or robot computer

39101: Drive System Error
Temperature too high on DC-link,
drive system %.0f.
Check:
Check cooling fan(s)
Check AC voltage to DC-link
Modify user program
Replace DC-link

39102: Drive System Error
Shunt resistor overload,
drive system %.0f.
Check:
Too much deceleration
Modify user program
Check AC voltage to DC-link
Replace DC-link

39103: Drive System Error
DC-link incoming mains not valid,
drive system %.0f.
Check:
Check voltage from Motor On contactor
Replace DC-link

39104: Drive System Error
DC-link voltage NOT OK,
drive system %.0f.
Check:
Check voltage from Motor On contactor.
Replace DC-link.

39105: Drive System Error
+/- 15V out of limit on DC-link,
drive system %.0f.
Check:
Check +/- 15V from power supply.
Replace DC-link.

39110: Drive System Error
Unknown type code for dc link
%s. Read typecode %d when
expecting %d.
Check:
Replace dc link.
Check serial link.

39111: Drive System Error
Wrong type detected for dc link
%s. %s found when
expecting %s.
Check:
Check/modify configuration.
Replace dc link.

39201: Drive System Error
Reset from drive unit,
joint: %s.
Check:
Restart start controller
Replace drive unit

39202: Drive System Error
Watchdog reset from drive unit,
joint: %s.
Check:
Restart start controller.
Replace drive unit.

39203: Drive System Error
+/- 15V out of limit on drive unit,
joint: %s.
Check:
Check +/- 15V from power supply.
Replace drive unit.

39204: Drive System Error
Too many consecutive communication/
synchronization errors on drive unit,
joint: %s.
Check:
Restart start controller.
Replace drive unit.
Replace drive unit right to the one
reporting the error.
Replace Robot Computer Board.

39205: Drive System Error
Internal HardWare/SoftWare error on
drive unit, joint: %s.
Internal error code: %s
Check:
Restart start controller.
Replace drive unit.

39206: Drive System Error
Glitch on short circuit detector on drive unit, joint: %s.

39207: Drive System Error
Short circuit detected on drive unit, joint: %s.

39208: Drive System Error
High temperature WARNING on drive unit, joint: %s.

39209: Drive System Error
High temperature ALARM on drive unit, joint: %s.
Check:
Allow system to cool down before restart.

39210: Drive System Error
Over temperature on drive unit, joint: %s.
Check:
Allow system to cool down before restart.

39211: Drive System Error
High temperature on transistors on drive unit, joint: %s.
Check:
Allow system to cool down before restart.

39212: Drive System Error
DC link voltage higher than allowed detected by drive unit,
joint: %s.
DC link voltage: %s (V)
Check:
Check inkomming mains.
Check/replace shunt resistors.
Check/replace DC-link.

39213: Drive System Error
Critical over voltage on DC link detected by drive unit,
joint: %s.
DC link voltage: %s (V)
Check:
Check inkomming mains.
Check/replace shunt resistors.
Check/replace DC-link.

39214: Drive System Error
Low DC voltage detected by drive unit, joint: %s. Voltage: %s (V).
NOTE !! This error will be disabled until next MOTOR ON.
Check:
Check inkomming mains.
Check program. Check konfiguration.
Check/replace DC-link.

39215: Drive System Error
Too big difference in 3 consecutive current references to drive unit,
joint: %s.
Check:
Restart system.
Check/replace resolver.
Check serial link(drivesystem

39216: Drive System Error
Too big difference in 3 consecutive rotor positions to drive unit,
joint: %s.
Check:
Restart system.
Check/replace resolver.
Check serial link(drivesystem

39217: Drive System Error
Current error: PWM ratio bigger than allowed. Drive unit, joint: %s.
DC link voltage: %s (V)
Check:
Check DC-voltage.
Check configuration.
Check motor/cables.

39218: Drive System Error
Broken motor cabling detected by drive unit, joint: %s.
DC link voltage: %s (V)
Check:
Check motor/cables.

39219: Drive System Error
Current error: Torque producing current lower than ordered. Drive unit, joint: %s.
DC link voltage: %s (V)
Check:
Check DC-voltage.
Check configuration.
Check motor/cables.

39220: Drive System Error
Current error: Torque producing current higher than ordered. Drive unit, joint: %s.
DC link voltage: %s (V)
Check:
Check configuration.
Check motor/cables.

39221: Drive System Error
Current error: Non torque producing current bigger than allowed.
Drive unit, joint: %s.
DC link voltage:%s (V)
Check:
Check DC-voltage.
Check configuration.
Check motor/cables.

39222: Drive System Error
Current error: The current is higher than max measurable due to short cirquit or unstabile current controller.
Drive unit, joint: %s.
DC link voltage:%s (V)
Check:
Check motor/cables (short cirquit).
Check configuration (motor parameters).
Replace drive unit.

39230: Drive System Error
Unknown drive unit type code, joint %s. Read typecode %d when expecting %d.
Check:
Replace drive unit.
Check serial link.

39231: Drive System Error
Wrong program revision in drive unit for joint %s.
Read revision (%d) out of range.
Check:
Replace drive unit.

39232: Drive System Error
Drive unit CPU, for joint %s did not start.
Check:
Replace drive unit.

39233: Drive System Error
Axis computer has detected too many consecutive communication errors from serial link to drive system %d.
Accumulated errors since warmstart: %d.
Check:
Restart start controller.
Check serial link.

39234: Drive System Error
Axis computer has detected transmission error from serial link to drive system %d.
Accumulated errors since warmstart: %d.
Check:
Check serial link.

39235: Drive System Error
Axis computer has detected a checksum error when transmitting parameters to drive unit for joint %s.
Check:
Restart controller.
Replace Drive unit.
Replace robot computer.

39236: Drive System Error
Wrong drive unit type detected for joint %s. %s found when expecting %s.
Check:
Check/modify configuration.
Replace drive unit.

39237: Drive System Error
Axis computer has detected too many communication absent errors from serial link to drive system %d.
Errors since warmstart: %d.
%d absent transmission of %d detected.
Check:
Restart start controller.
Check serial link.

39238: Drive System Error
Axis computer has detected transmission absent error from serial link to drive system %d.
Accumulated errors since warmstart: %d.
Check:
Check serial link.

39301: External Drive Error
High DC voltage detected by external drive unit,
joint: %s.
Check:
See documentation for Atlas DMC/FBU.
Check inkomming mains.
Check/replace shunt resistors.
Check/replace External Drive Unit.

39302: External Drive Error
High motor temperature detected by drive unit,
joint: %s.
Check:
Allow motor to cool down.
Check/replace External Drive Unit.
See documentation for Atlas DMC/FBU.

39303: External Drive Error
High Power device temperature on external drive unit,
joint: %s.
Check:
Allow drive unit to cool down.
Check/replace External Drive Unit.
See documentation for Atlas DMC/FBU.

39304: External Drive Error
Current regulator fault detected on external drive unit,
joint: %s.
Check:
See documentation for Atlas DMC/FBU.
Check/replace External Drive Unit.

39305: External Drive Error
Short circuit detected on drive unit,
joint: %s.
Check:
See documentation for Atlas DMC/FBU.
Check/replace External Drive Unit.

39306: External Drive Error
Missing ENABLE for external drive unit, joint: %s.
Check:
See documentation for Atlas DMC/FBU.

39307: External Drive Error
External drive unit, joint:
%s has performed
a CPU reset.
Check:
Restart the system.
See documentation for Atlas DMC/FBU.
Check/replace External Drive Unit.

39308: External Drive Error
Too many consecutive communication errors on external drive unit,
joint: %s.
Check:
Restart start controller.
Replace drive unit.
Replace drive unit right to the one reporting the error.
Replace Robot Computer Board.

39309: External Drive Error
Too many consecutive synchronization errors on external drive unit, joint: %s.
Check:
Restart start controller.
Replace external drive unit.
Replace Robot Computer Board.

39310: External Drive Stop
External drive, joint: %s has order a program stop.
Check:
See documentation for Atlas DMC/ FBU.

39311: External Drive EM-Stop
External drive, joint: %s has order an emergancy stop.
Check:
See documentation for Atlas DMC/ FBU.

39320: External drive error
Time out in communication with external drive unit, joint: %s
Check:
Restart system and external drive
Replace external drive
See documentation for Atlas DMC/ FBU.

4 Program error messages

40001: Argument error
Task %.16s: More than one occurence of optional parameter %.16s
Check:
Make sure that the optional parameter is not specified more than once in the same routine call.

40002: Argument error
Task %.16s: Excluding arguments must have conditional value (%.16s has value)
Check:
Arguments may not be specified for more than one parameter from a list of parameters that exclude each other unless all values are conditional argument values.

40003: Argument error
Task %.16s: Expecting argument for required parameter %.16s but found optional argument %.16s
Check:
Check that the arguments are specified in the same order as the parameters for the routine being called.

40004: Argument error
Task %.16s: Argument for REF parameter %.16s is not data reference
Check:
Make sure the argument expression is just a data or parameter reference.

40005: Argument error
Task %.16s: Argument for ’INOUT’ parameter %.16s is not variable or persistent reference or is read only.
Check:
Make sure the argument is just a variable, persistent, variable parameter or persistent parameter reference and it is writeable.
Do not use () around the argument.

40006: Argument error
Task %.16s: Missing optional argument value for parameter %.16s
Check:
Only ’switch’ parameters may be specified by name only. Optional parameters of other types must be assigned a value. Add a value.

40007: Argument error
Task %.16s: Optional argument
%.16s at wrong place in
argument list
Check:
Check that the arguments are specified in the same order as the parameters for the routine being called.

40008: Argument error
Task %.16s: Reference to
optional parameter %.16s in
required argument
Check:
An argument corresponding to an optional
parameter must be specified with a leading '\'character. Change the required argument into an optional.

40009: Argument error
Task %.16s: Reference to
required parameter %.16s in
conditional argument value
Check:
A conditional value for an optional parameter must refer an optional parameter in the calling routine. Change the conditional value.

40010: Argument error
Task %.16s: Reference to
required parameter %.16s in
optional argument
Check:
An argument corresponding to a required parameter must not be specified with the leading '\' character. Change the optional argument into a reguired.

40011: Argument error
Task %.16s: Named required
argument %.16s at wrong place
in argument list
Check:
Check that the arguments are specified in the same order as the parameters for the routine being called.

40012: Argument error
Task %.16s: 'switch' argument
%.16s cannot have a value
Check:
An argument corresponding to a 'switch' parameter may not be assigned a value. Remove the value.

40013: Argument error
Task %.16s: Too few arguments
in call to routine %.16s
Check:
A routine call must supply values for all required parameters of the routine being called. Add more arguments to fit the parameter list.

40014: Argument error
Task %.16s: Too many
arguments in call to routine
%.16s
Check:
Remove arguments so that no arguments are supplied in excess to those defined by the parameter list of the called routine.

40015: Data declaration error
Task %.16s: Array dimension
must be > 0 (value is %i)
Check:
Array dimensions must be positive. Change the dimension expression.

40016: Data declaration error
Task %.16s: Too many
dimensions in array definition
Check:
An array may have at most 3 dimensions. Rewrite the program so that no more than 3 dimensions are needed.

40017: Type error
Task %.16s: Indexed data
%.18s %.18s is
not of array type
Check:
Only data that have been declared to be arrays may be indexed. Remove the index or indices, or declare the data to be an array.

40018: Type error
Task %.16s: Data
%.18s %.18s is
not of record type
Check:
Components are only available for data of record type. Check the type and name of the referenced data.

40019: Limit error
Task %.16s: Error when
creating the persistent variable
(internal error code %i)
%.16s
Check:
An error occurred when the persistent was to be inserted into the shared database. Probably the database is full. Ref. to system parameter AveragePers.

40020: Data declaration error
Task %.16s: Expression not
constant expression (%.16s
not constant)
Check:
Expressions contained within data declarations must be constant expressions. Make sure the expression does not contain any variable or persistent reference, or function call.

40021: Instruction error
Task %.16s: RETURN from
function must have an expression
Check:
A RETURN instruction within a function must specify a function value to be returned. Add a value expression.

40022: Type error
Task %.16s: Illegal
combination of operand types
%.18s and
%.18s for ‘*' operator
Check:
The allowed type combinations for the two operands of the ’*’ operator are ’num’*’num’, ’num’*’pos’, ’pos’*’num’, ’pos’*’pos’ and ’orient’*’orient’. Check the types of the operands.

40023: Instruction error
Task %.16s: Cannot transfer
control into another instruction list
Check:
Make sure that the label is located in the same instruction list as the GOTO instruction, at the same or an outer level. It is not possible to jump into a program flow instruction.

40024: Type error
Task %.16s: Illegal type
%.18s for left operand of
binary ’+’ or ’-’ operator
Check:
The allowed types for the operands of the ’+’ operator are ’num’, ’pos’ and ’string’, for the ’-’ operator ’num’ and ’pos’. Check the type of the operand.

40025: Type error
Task %.16s: Illegal type
%.18s for operand of unary
’+’ or ’-’ operator
Check:
The allowed types for the operands of the ’+’ and ’-’ operators are ’num’ and ’pos’. Check the type of the operand.

40026: Type error
Task %.16s: Illegal type
%.18s for right operand of
binary ’+’ or ’-’ operator
Check:
The allowed types for the operands of the ’+’ operator are ’num’, ’pos’ and ’string’, for the ’-’ operator ’num’ and ’pos’. Check the type of the operand.

40027: Type error
Task %.16s: Illegal type
%.18s for left operand of
’/’, ’DIV’ or ’MOD’ operator
Check:
The only allowed type for the operands of the ’/’, ’DIV’ and ’MOD’ operators is ’num’. Check the type of the operand.

40028: Type error
Task %.16s: Illegal type
%.18s for right operand of
’/’, ’DIV’ or ’MOD’ operator
Check:
The only allowed type for the operands of the ’/’, ’DIV’ and ’MOD’ operators is ’num’. Check the type of the operand.

40029: Type error
Task %.16s: Illegal type
%.18s for left operand of
’<’, ’<=’, ’>’ or ’>=’ operator
Check:
The only allowed type for the operands of the ’<’, ’<=’, ’>’ and ’>=’ operators is ’num’. Check the type of the operand.

40030: Type error
Task %.16s: Illegal type
%.18s for right operand of
’<’, ’<=’, ’>’ or ’>=’ operator
Check:
The only allowed type for the operands of the ’<’, ’<=’, ’>’ and ’>=’ operators is ’num’. Check the type of the operand.

40031: Type error
Task %.16s: Illegal type
%.18s for left operand of
’*’ operator
Check:
The allowed types for the operands of the ’*’ operator are ’num’, ’pos’ and ’orient’. Check the type of the operand.

40032: Type error
Task %.16s: Illegal type
%.18s for right operand of
’*’ operator
Check:
The allowed types for the operands of the ’*’ operator are ’num’, ’pos’ and ’orient’. Check the type of the operand.

40033: Type error
Task %.16s: Illegal type
%.18s for operand of ’NOT’
operator
Check:
The only allowed type for the operand of the ’NOT’ operator is ’bool’. Check the type of the operand.

40034: Type error
Task %.16s: Illegal type
%.18s for left operand of
’OR’, ’XOR’ or ’AND’ operator
Check:
The only allowed type for the operands of the ’OR’, ’XOR’ and "AND’ operator is ’bool’. Check the type of the operand.

40035: Type error
Task %.16s: Illegal type
%.18s for right operand of
’OR’, ’XOR’ or ’AND’ operator
Check:
The only allowed type for the operands of the ’OR’, ’XOR’ and "AND’ operator is ’bool’. Check the type of the operand.

40036: Type error
Task %.16s: Incorrect number
of indices in index list for array
%.18s with %i dimension(s)
Check:
Make sure that the number of indices in the index list is the same as the number of dimensions of the indexed data array.

40037: Data declaration error
Task %.16s: LOCAL illegal in
routine constant declaration
Check:
Only program data declarations may have the LOCAL attribute. Remove the LOCAL attribute or move the declaration outside of the routine.

40038: Data declaration error
Task %.16s: LOCAL illegal in
routine variable declaration
Check:
Only program data declarations may have the LOCAL attribute. Remove the LOCAL attribute or move the declaration outside of the routine.

40039: Name error
Task %.16s: Constant name
%.16s ambiguous
Check:
Routine data must have names that are unique within the routine. Program data must have names that are unique within the module. Rename the data or change the conflicting name.

40040: Name error
Task %.16s: Global constant
name %.16s ambiguous
Check:
Global data must have names that are unique among all the global types, data, global routines and modules in the entire program. Rename the data or change the conflicting name.

40041: Name error
Task %.16s: Global persistent
name %.16s ambiguous
Check:
Global data must have names that are unique among all the global types, data, global routines and modules in the entire program. Rename the data or change the conflicting name.

40042: Name error
Task %.16s: Global routine
name %.16s ambiguous
Check:
Global routines must have names that are unique among all the global types, data, global routines and modules in the entire program. Rename the routine or change the conflicting name.

40043: Name error
Task %.16s: Global variable
name %.16s ambiguous
Check:
Global data must have names that are unique among all the global types, data, global routines and modules in the entire program. Rename the data or change the conflicting name.

40044: Name error
Task %.16s: Label name
%.16s ambiguous
Check:
Labels must have names that are unique within the routine. Rename the label or change the conflicting name.

40045: Name error
Task %.16s: Module name
%.16s ambiguous
Check:
Modules must have names that are unique among all the global types, global data, global routines and modules in the entire program. Rename the module or change the conflicting name.

40046: Name error
Task %.16s: Parameter name
%.16s ambiguous
Check:
Parameters must have names that are unique within the routine. Rename the parameter or change the conflicting name.

40047: Name error
Task %.16s: Persistent name
%.16s ambiguous
Check:
Program data must have names that are unique within the module. Rename the data or change the conflicting name.

40048: Name error
Task %.16s: Routine name
%.16s ambiguous
Check:
Routines must have names that are unique within the module. Rename the routine or change the conflicting name.

40049: Name error
Task %.16s: Variable name
%.16s ambiguous
Check:
Routine data must have names that are unique within the routine. Program data must have names that are unique within the module. Rename the data or change the conflicting name.

40050: Type error
Task %.16s: Operand types
%.18s and
%.18s for binary ’+’ or ’-’
operator not equal
Check:
The two operands of the ’+’ and ’-’ operators must have equal type. Check the operand types.

40051: Type error
Task %.16s: Operand types
%.18s and
%.18s for ’=’ or ’<>’
operator not equal
Check:
The two operands of the ’=’ and ’<>’ operators must have equal type. Check the operand types.

40052: Instruction error
Task %.16s: RETURN with
expression only allowed in function
Check:
In a procedure or trap the RETURN instruction must not specify a return value expression. Remove the expression.

40053: Instruction error
Task %.16s: RAISE in error
handler must not have an expression
Check:
A RAISE instruction within an error handler can only be used to propagate the current error, and may therefore not specify an error number. Remove the error number expression.

40054: Type error
Task %.16s: Different
dimension of array type (%i) and
aggregate (%i)
Check:
Make sure that the number of expressions in the aggregate is the same as the dimension of the data array.

40055: Type error
Task %.16s: Assignment target
type %.18s is not value or
semi-value type
Check:
The type, of the data to be assigned a value, must be a value or semi-value type. Data of non-value types may only be set by special type specific predefined instructions or functions.

40056: Type error
Task %.16s: Type
%.18s for left operand of
'=' or '<>' operator not value or
semi-value type
Check:
The '=' and '<>' operators may only be applied to expressions of value or semi-value type. If comparisons are to be made, special type specific predefined functions are needed.

40057: Type error
Task %.16s: Type
%.18s for right operand of
'=' or '<>' operator not value or
semi-value type
Check:
The '=' and '<>' operators may only be applied to expressions of value or semi-value type. If comparisons are to be made, special type specific predefined functions are needed.

40058: Type error
Task %.16s: TEST expression
type %.18s not value or
semi-value type
Check:
The TEST instruction may only be applied
to an expression of value or semi-value type. If comparisons are to be made, special type specific predefined functions are needed.

40059: Data declaration error
Task %.16s: Place holder for
value expression not allowed in
definition of named constant
Check:
Complete the data declaration or change the data name to a place holder.

40060: Data declaration error
Task %.16s: Place holder for
array dimension not allowed in
definition of named constant or variable
Check:
Complete the data declaration or change the data name to a place holder.

40061: Routine declaration error
Task %.16s: Place holder for
parameter array dimensions not allowed
in definition of named routine
Check:
Complete the parameter declaration or change the routine name to a place holder.

40062: Name error
Task %.16s: Place holder for
parameter name not allowed in definition
of named routine
Check:
Complete the routine declaration or change the routine name to a place holder.

40063: Data declaration error
Task %.16s: Place holder for
initial value expression not allowed in
definition of named persistent
Check:
Complete the data declaration or change the data name to a place holder.

40064: Routine declaration error
Task %.16s: Place holder for
parameter not allowed in definition of
named routine
Check:
Complete the parameter declaration, remove the place holder or change the routine name to a place holder.

40065: Reference error
Task %.16s: Place holder for
type not allowed in definition of named data, record component or routine
Check:
Complete the data or routine declaration or change the data or routine name to a place holder.

40066: Data declaration error
Task %.16s: Place holder for
initial value expression not allowed in definition of named variable
Check:
Complete the data declaration or change the data name to a place holder.

40067: Type error
Task %.16s: Too few
components in record aggregate of type %.18s
Check:
Make sure that the number of expressions in the aggregate is the same as the number of components in the record type.

40068: Type error
Task %.16s: Too many
components in record aggregate of type %.18s
Check:
Make sure that the number of expressions in the aggregate is the same as the number of components in the record type.

40069: Reference error
Task %.16s: Data reference
%.16s is ambiguous
Check:
At least one other object sharing the same name as the referred data is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40070: Reference error
Task %.16s: Function
reference %.16s is ambiguous
Check:
At least one other object sharing the same name as the referred function is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40071: Reference error
Task %.16s: Label reference
%.16s is ambiguous
Check:
At least one other object sharing the same name as the referred label is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40072: Reference error
Task %.16s: Procedure
reference %.16s is ambiguous
Check:
At least one other object sharing the same name as the referred procedure is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40073: Reference error
Task %.16s: Trap reference
%.16s is ambiguous
Check:
At least one other object sharing the same name as the referred trap is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40074: Reference error
Task %.16s: %.16s
not entire data reference
Check:
The specified name identifies an object other than data. Check if the desired data is hidden by some other object with the same name.

40075: Reference error
Task %.16s: %.16s
not function reference
Check:
The specified name identifies an object other than a function. Check if the desired function is hidden by some other object with the same name.

40076: Reference error
Task %.16s: %.16s
not label reference
Check:
The specified name identifies an object other than a label. Check if the desired label is hidden by some other object with the same name.

40077: Reference error
Task %.16s: %.16s
not optional parameter reference in conditional argument value
Check:
The specified name identifies an object other than an optional parameter. Change the name to refer to an optional parameter.

40078: Reference error
Task %.16s: %.16s
not optional parameter reference
Check:
The specified name identifies an object other than an optional parameter. Change the name to refer to an optional parameter.

40079: Reference error
Task %.16s: %.16s
not procedure reference
Check:
The specified name identifies an object other than a procedure. Check if the desired procedure is hidden by some other object with the same name.

40080: Reference error
Task %.16s: %.16s
not required parameter reference
Check:
The specified name identifies an object other than a required parameter. Change the name to refer to a required parameter.

40081: Reference error
Task %.16s: %.16s
not trap reference
Check:
The specified name identifies an object other than a trap. Check if the desired trap is hidden by some other object with the same name.

40082: Reference error
Task %.16s: %.16s
not type name
Check:
The specified name identifies an object other than a type. Check if the desired type is hidden by some other object with the same name.

40083: Type error
Task %.16s: %.16s
not value type
Check:
Only variables which lack initial value, and ’VAR’ mode parameters may be of semi-value or non-value type.

40086: Reference error
Task %.16s: Reference to unknown label %.16s
Check:
The routine contains no label (or other object) with the specified name.

40087: Reference error
Task %.16s: Reference to unknown optional parameter %.16s
Check:
The called routine contains no optional parameter (or other object) with the specified name.

40089: Reference error
Task %.16s: Reference to
unknown record component
%.16s
Check:
The record type contains no record component with the specified name.

40090: Reference error
Task %.16s: Reference to
unknown required parameter
%.16s
Check:
The called routine contains no required parameter (or other object) with the specified name.

40092: Reference error
Task %.16s: Unknown type name
%.16s
Check:
No data type (or other object) with the specified name is visible from this program position.

40093: Instruction error
Task %.16s: Assignment target
is read only
Check:
The data to be assigned a value may not be a constant, read only variable or read only persistent.

40094: Data declaration error
Task %.16s: Persistent
declaration not allowed in routine
Check:
Persistents may only be declared at module level. Move the persistent declaration from the routine.

40095: Instruction error
Task %.16s: RAISE without
expression only allowed in error handler
Check:
Add an error number expression to the RAISE instruction.

40096: Instruction error
Task %.16s: RETRY only
allowed in error handler
Check:
The RETRY instruction may only be used in error handlers. Remove it.

40097: Instruction error
Task %.16s: TRYNEXT only
allowed in error handler
Check:
The TRYNEXT instruction may only be used in error handlers. Remove it.

40098: Parameter error
Task %.16s: 'switch'
parameter must have transfer mode IN
Check:
Remove the parameter transfer mode specifier. If IN transfer mode is not sufficient, change the data type of the parameter.

40099: Parameter error
Task %.16s: 'switch'
parameter cannot be dimensioned
Check:
Remove the array dimension specification, or change the data type of the parameter.

40100: Parameter error
Task %.16s: 'switch' only
allowed for optional parameter
Check:
Change the parameter into an optional parameter, or change the data type of the parameter. If the object is not a parameter, change the data type.

40101: Type error
Task %.16s: Type mismatch of
expected type %.18s and
type %.18s
Check:
The expression is not of the expected data type.

40102: Type error
Task %.16s: Type mismatch of aggregate, expected type %.18s
Check:
The aggregate does not match the expected data type.

40103: Type error
Task %.16s: Persistent %.18s %.16s type mismatch
Check:
There is already a persistent data with the same name but with another data type. Rename the persistent, or change its data type.

40104: Data declaration error
Task %.16s: Cannot determine array dimensions (circular constant references ?)
Check:
Check that any referred constants are correctly defined. If so, the program is too complex. Try to rewrite the declarations.

40105: Data declaration error
Task %.16s: Cannot determine type of constant value (circular constant references ?)
Check:
Check that any referred constants are correctly defined. If so, the program is too complex. Try to rewrite the declarations.

40106: Data declaration error
Task %.16s: Cannot evaluate constant value expression (circular constant references ?)
Check:
Check that any referred constants are correctly defined. If so, the program is too complex. Try to rewrite the declarations.

40107: Data declaration error
Task %.16s: Cannot determine type of variable value (circular constant references?)
Check:
Check that any referred constants are correctly defined. If so, the program is too complex. Try to rewrite the declarations.

40108: Type error
Task %.16s: Unknown aggregate type
Check:
An aggregate may not be used in this position since there is no expected data type. Declare data with the desired data type and aggregate value. Use the name of the data instead of the aggregate.

40109: Type definition error
Task %.16s: Cannot determine type of record component %.16s
(circular type definitions?)
Check:
Check that the type of the component is correctly defined. If so, it could be a circular definition, the type of a component could not refere to the its own record type.

40110: Reference error
Task %.16s: Record name %.16s is ambiguous
Check:
At least one other object sharing the same name as the referred record name is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40111: Name error
Task %.16s: Global record name %.16s ambiguous
Check:
Global type must have names that are unique among all the global types, data, global routines and modules in the entire program. Rename the record or change the conflicting name.

40112: Reference error
Task %.16s: Alias name
%.16s is ambiguous
Check:
At least one other object sharing the same name as the referred alias name is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40113: Name error
Task %.16s: Global alias
name %.16s ambiguous
Check:
Global type must have names that are unique among all the global types, data, global routines and modules in the entire program. Rename the alias or change the conflicting name.

40114: Type definition error
Task %.16s: Type reference
of alias name %.16s
is an alias type
Check:
Check that the type of the component is correctly defined. If so, it could be a circular definition, the type of a component could not refere to the its own record type.

40115: Type definition error
Task %.16s: Cannot determine
type of alias %.16s
(circular type definitions?)
Check:
Check that the type of the alias is correctly defined. If so, it could be a circular definition, the type of an alias could not refere to a record that use this alias as a component.

40116: Reference error
Task %.16s:
Record component name
%.16s is ambiguous
Check:
At least one other object sharing the same name as the referred component is visible from this program position. Make sure that all object names fulfill the naming rules regarding uniqueness.

40117: Type definition error
Task %.16s: Place holder for
record component not allowed in
definition of named record
Check:
Complete the definition or change the data name to a place holder.

40118: Not authorized
Task %.16s: The function:
User defined data types is not
installed in this system
Check:
Install the option Developer Functions in the system.

40119: Reference error
Task %.16s:
Cannot use the semi-value type
%.16s for record components

40120: Reference error
Task %.16s: Illegal reference
to installed task object
%.16s from shared object
Check:
Install the referred object shared, or install the referring ReaL object/ archive or RAPID module in each task (not shared

40141: Argument error
Task %.16s: Argument for
'PERS' parameter %.16s is not
persistent reference or is read only
Check:
Make sure the argument is just a persistent or persistent parameter reference and it is writeable.
Do not use () around the argument.

40142: Argument error
Task %.16s: Argument for
'VAR' parameter %.16s is not
variable reference or is read only
Check:
Make sure the argument is just
a variable or variable parameter
reference and it is writeable.
Do not use () around the argument.

40157: Instruction error
Task %.16s: Interrupt number
is not static variable reference, is
shared, or is read only
Check:
Make sure the interrupt number is just
a variable or variable parameter
reference. The variable must be
static and not shared. The variable may
not be read only.

40158: Value error
Task %.16s: Integer value
%G too large
Check:
The value of the expression must be an
integer value. The current value is
outside the integer range.

40159: Value error
Task %.16s: %G not
integer value
Check:
The value of the expression must be an
exact integer value. The current value
has a fraction part.

40165: Reference error
Task %.16s: Reference to
unknown entire data %.16s
Check:
No data (or other object) with the
specified name is visible from this
program position.

40166: Reference error
Task %.16s: Reference to
unknown function %.16s
Check:
No function (or other object) with the
specified name is visible from this
program position.

40168: Reference error
Task %.16s: Reference to
unknown procedure %.16s
Check:
No procedure (or other object) with the
specified name is visible from this
program position.

40170: Reference error
Task %.16s: Reference to
unknown trap %.16s
Check:
No trap (or other object) with the
specified name is visible from this
program position.

40191: Instruction error
Task %.16s: Variable and trap
routine already connected
Check:
It is not legal to connect a specific
variable with a specific trap routine
more than once.

40192: Argument error
Task %.16s: %.16s
is second present conditional argument
for excluding parameters
Check:
Arguments may not be present for more
than one parameter from a list of
parameters that exclude each other.

40193: Execution error
Task %.16s: Late binding
procedure call error %i
Check:
There is an error in the procedure call
instruction. See previous message for
the actual cause.

40194: Value error
Task %.16s: Division by zero
Check:
Cannot divide by 0. Rewrite the program so that the divide operation is not executed when the divisor is 0.

40195: Limit error
Task %.16s: Exceeded maximum number %i of allowed RETRYs
Check:
The error correction performed before the RETRY instruction is executed, is probably not enough to cure the error. Check the error handler.

40196: Instruction error
Task %.16s: Attempt to execute place holder
Check:
Remove the place holder or the instruction containing it, or make the instruction complete. Then continue execution.

40197: Execution error
Task %.16s: Function does not return any value
Check:
The end of the function has been reached without a RETURN instruction being executed. Add a RETURN instruction specifying a function return value.

40198: Value error
Task %.16s: Illegal orientation value %.40s
Check:
Attempt to use illegal orientation (quaternion) value

40199: Value error
Task %.16s: Illegal error number %i in RAISE
Check:
Only error numbers in the range 1-99 are allowed in the RAISE instruction.

40200: Limit error
Task %.16s: No more interrupt number available
Check:
There is a limited number of interrupt numbers available. Rewrite the program to use fewer interrupt numbers. This message may also occur as a consequence of a system error.

40201: Value error
Task %.16s: Negative operand %i not allowed
Check:
The ’MOD’ operator only allows non negative operands. Change the program to make sure that the operator is not applied to negative values.

40202: Type error
Task %.16s: Dimensions %i and %i of conformant array dimension number %i are incompatible
Check:
The array is not of the expected size. Array assignment may only be performed on arrays of identical size.

40203: Reference error
Task %.16s: Optional parameter %.16s not present
Check:
The value of a non present optional parameter may not be referred. Use the predefined function ’Present’ to check the presence of the parameter before using its value.

40204: Value error
Task %.16s: Array index %i for dimension number %i out of bounds (1-%i)
Check:
The array index value is non-positive or violates the declared size of the array.

40205: Value error
Task %.16s: String too long
Check:
String value exceeds the maximum allowed
length. Rewrite the program to use strings of lesser length.

40221: Execution error
Task %.16s: Execution aborted
Check:
Execution was aborted due to a fatal error.

40222: Limit error
Task %.16s: Execution stack overflow
Check:
The program is too complex to execute. Probably the program contains recursive routines.

40223: Execution error
Task %.16s: Fatal runtime error
Check:
A fatal runtime error has occurred. Fatal runtime errors causes immediate termination of execution. See previous message for the actual cause.

40224: Execution error
Task %.16s: Illegal return code %i from ReaL routine
Check:
This is always caused by an internal error in the ReaL routine.

40225: Execution error
Task %.16s: Execution could not be restarted
Check:
Execution could not be continued after power failure. Restart the program.

40226: Name error
Task %.16s: Procedure name
%.40s
is not a RAPID identifier excluding reserved words
Check:
The procedure name, must be a legal RAPID identifier not equal to any of the reserved words of the RAPID language. Change the name expression.

40227: Limit error
Task %.16s: Runtime stack overflow
Check:
The program is too complex to execute. Probably the program contains recursive routines.

40228: Execution error
Task %.16s: Unhandled non-fatal runtime error %i
Check:
A non-fatal runtime error has occurred but was not handled by any ERROR clause. See previous message for the actual cause.

40229: Execution error
Task %.16s: Unhandled raise error %i
Check:
An error was raised by a RAISE instruction but was not handled by any ERROR clause.

40230: Execution error
Task %.16s: Unhandled non-fatal runtime error
Check:
A non-fatal runtime error has occurred but was not handled by any ERROR clause.

40241: Value error
Task %.16s: Array dimension number %G out of range (1-%i)
Check:
The value of the ’DimNo’ parameter of the ’Dim’ function must be an integer value in the specified range.

40242: Type error
Task %.16s: Data is not an
array
Check:
The ’DatObj’ parameter of the ’Dim’ function must be an array.

40243: Value error
Task %.16s: Unknown interrupt
number
Check:
Check that the specified interrupt variable has been initialized by CONNECT, and that the interrupt has
been
defined using the ISignalDI or other interrupt definition instruction.

40244: Value error
Object %.16s is of non-value
type
Check:
Use expression or data object of value or semivalue type.

40245: Parameter error
Parameters in %.16s and %.16s is not matching (late binding)
Check:
Make sure that all procedures that are called from the same late binding node have matching parameters. I.e they should be matching concerning base type,
mode and required/optional parameters.

40251: Name error
Task %.16s: Ambiguous symbol
name %.16s
Check:
Installed objects must have names that are unique. Rename the object or change the conflicting name.

40252: Limit error
Task %.16s: Error %i when
creating sdb entry for %.16s
Check:
An error occurred when the persistent was to be inserted into the shared database. Probably the database is full.

40253: Type definition error
Task %.16s: Alias
%.16s of alias
%.16s not allowed
Check:
Is is not possible to define an alias type equal to another alias type. Instead, define two alias types equal to the same atomic or record type.

40254: Symbol definition error
Task %.16s: ’ANYTYPE#’
parameter %.16s cannot be
dimensioned
Check:
Remove the dimension specification. ’ANYTYPE#’ includes array types.

40255: Symbol definition error
Task %.16s: ’ANYTYPE#’ only
allowed for parameter (not for
%.16s)
Check:
Use another type.

40256: Parameter error
Task %.16s: ’alt’ must not be
set for first optional parameter
%.16s in alternatives list
Check:
Make sure that only the second and following in each list of excluding optional parameters are marked as alternatives.

40257: Parameter error
Task %.16s: REF mode
parameter %.16s cannot be
dimensioned
Check:
Remove the array dimension specification, or change the mode of the parameter.

40258: Parameter error
Task %.16s: ’switch’
parameter %.16s can not be
dimensioned
Check:
Remove the array dimension specification, or change the data type of the parameter.

40259: Parameter error
Task %.16s: ’switch’
parameter %.16s must have
transfer mode IN (specified value
%i)
Check:
Remove the parameter transfer mode specifier. If IN transfer mode is not sufficient, change the data type of the parameter.

40260: Symbol definition error
Task %.16s: ’switch’ only
allowed for optional parameter
(not for %.16s)
Check:
Change the parameter into an optional parameter, or change the data type of the parameter. If the object is not a parameter, change the data type.

40261: Type definition error
Task %.16s: Value type class
for %.16s must be one of
REAL_SYMVALTYP_VAL,
_SEMIVAL, _NONVAL or
_NONE (specified value %i)
Check:
Change the value type class.

40262: Data declaration error
Task %.16s: Too many array
dimensions for %.16s
(specified value %i)
Check:
An array may have at most 3 dimensions.

40263: Name error
Task %.16s: Symbol name
%.40s
is not a RAPID identifier excluding
reserved words
Check:
The names of installed objects, including parameters and components, must be legal RAPID identifiers not equal to any of the reserved words of the RAPID language. Change the name.

40264: Symbol definition error
Task %.16s: Missing C
function for %.16s
Check:
A C-function that executes the ReaL function being defined, must be specified.

40265: Symbol definition error
Task %.16s: Missing value
initialization function for
%.16s
Check:
A value initialization function must be specified.

40266: Reference error
Task %.16s: %.16s
is not a data type name (object
%.16s)
Check:
The specified name identifies an object other than a type.

40267: Reference error
Task %.16s: %.16s
is not a value data type (object
%.16s)
Check:
Only record components, alias types, variables and ’VAR’ mode parameters may
be of semi-value or non-value type.

40268: Symbol definition error
Task %.16s: Missing value
conversion function for %.16s
Check:
A value conversion function must be specified for a semi-value type.

40269: Symbol definition error
Task %.16s: Not enough
memory for value of data
%.16s
Check:
More memory required.

40270: Type definition error
Task %.16s: Private type
%.16s can only be semi-value
or non-value type (specified value
%i)
Check:
Change the value type class.

40271: Type definition error
Task %.16s: Private type
%.16s size must be multiple
of 4 (specified value %i)
Check:
All RAPID types must have a size that is a multiple of four. Change the specified type size.

40272: Type error
Task %.16s: Persistent type
mismatch for %.16s
Check:
There is already a persistent data with the same name but with another data type. Rename the persistent, or change its data type.

40273: Reference error
Task %.16s: Unknown data type
name %.16s for
%.16s
Check:
There is no data type (or other object) with the specified name.

40274: Parameter error
Task %.16s: Unknown parameter
transfer mode %i for
%.16s
Check:
The specified parameter transfer mode is not one of IN, ’VAR’, ’PERS’, ’INOUT’ or REF. Use corresponding REAL_SYMPARMOD_x.

40275: Symbol definition error
Task %.16s: Unknown symbol
definition type %i
Check:
The symbol definition type tag does not specify one of the allowed symbol types (REAL_SYMDEF_x

40276: Symbol definition error
Task %.16s: Initialization
function may not be specified for shared variable %.16s
with per task value
Check:
Remove initialization function, install variable in all tasks, or make variable value shared.

40301: I/O error
Task %.16s: Permission denied
(file name
%.39s)

40302: I/O error
Task %.16s: No such file or
directory (file name
%.39s)

40303: I/O error
Task %.16s: No space left on
device (file name
%.39s)

40304: I/O error
Task %.16s: I/O error %!
(file name
%.39s)
Check:
One of:
Permission denied (write protected)
No such file or directory
No space left on device

40321: Load error
Task %.16s:
Module loaded with path
%.40s
is active
Check:
A module containing routines or data that are still active cannot be erased.

40322: Load error
Task %.16s: RAPID syntax
error(s) in file
%.40s
Check:
The source file to be loaded contains RAPID syntax errors. Correct the source file. The syntax errors are logged in a separate file.

40323: Load error
Task %.16s: Syntax error(s)
in header in file
%.40s
Check:
The source file to be loaded contains syntax error in the file header. Correct the source file. The syntax errors are logged in a separate file.

40324: Load error
Task %.16s: Keywords not
defined in specified language (file
%.39s)
Check:
Cannot load RAPID source code in the national language specified in the file header.

40325: Load error
Task %.16s: Not enough heap
space
Check:
There is not enough free memory left.

40326: Load error
Task %.16s: Parser stack
full (file
%.39s)
Check:
The program is too complex to load.

40327: Load error
Task %.16s: Not current RAPID
version (file
%.39s)
Check:
Cannot load RAPID source code of the version specified in the file header.

40351: Memory allocation error
Task %.16s: Failed to
allocate hash table, use linear list

40352: Memory allocation error
Task %.16s: Failed to
update persistent expression, keep old one

40501: Timeout
%s
%s

40502: Digital input break
%s
%s

40503: Reference error
Device descriptor is
%s

40504: Parameter error
%s

40505: File access error
%s
%s

40506: System access error
%s
%s

40507: Limit error
%s

40508: Wrong orientation value
in %s

40509: Search warning
%s
Before performing next search, make sure that TCP is moved back to the start position of the search path.
Check:
If no repositioning is done, before restart of circular search, movement that can cause damage might occur.

40510: Security warning
The move instruction can't restart due to security problem.
Try to move the PP

40511: Parameter error
The parameter %.16s in %.16s is specified with a negative value
Check:
The parameter must be set positive.

40512: Missing ext. axis value
Some active external axis have incorrect or no order value.
Reprogram the position.

40513: Mechanical unit error
Not possible to activate or deactivate mechanical unit.

40514: Execution error
Too far from path to perform StartMove of the interrupted movement.
Check:
Position the robot to the interrupted position in the program.

40515: Type error
Task %s: Illegal data type of argument for parameter %s

40516: Value error
Task %s: Illegal value of argument for parameter %s

40517: Search error
%s
No search hit or more than 1 search hit during stepwise forward execution. The search instruction is ready and next instruction can be executed.
Check:
Note that no position has been returned from the search instruction.

40518: Type error
%s

40590: ParId error
%.40s
%.40s
Check:
%.40s

40591: Argument error
Unknown type of parameter identification.

40592: ParId Program Stop
Any type of program stop during load identification is not allowed.
Check:
Start the identification procedure from beginning again.

40593: ParId Power Fail
Power Fail during load identification results in faulty load result.
Check:
Restart the program execution again with same run mode (without PP move) for load identification from beginnig.

40594: ParId user error
Error resulting in raise of PP to the beginning of the parameter identification procedure.
Check:
Start the identification procedure from beginning again.

40595: Argument error
Unknown type of load
identification.

40596: ParId Program Stop
Any type of program stop during
load identification is not allowed.
Check:
Restart the program execution again
for load identification from beginnig.
(From old PP pos. and with same run
mode
if try to move PP within program list

40597: ParId Speed Override
Speed override not 100 per cent.
Check:
Change the speed override to 100.
Restart the program execution again
for load identification from beginnig.

40600: Argument error
No WObj specified
for movement with stationary TCP.
Check:
Add argument WObj
for actual work object.
If not movement with stationary TCP,
change argument Tool to
"robot holds the tool

40601: Argument error
Undefined if robot holds the tool
or the work object.
Check:
Check if mismatch between
argument Tool and argument WObj
for data component robhold.

40602: Argument error
Argument %s has
at least one data component
with negative value.
Check:
Set all data components
in argument %s
to positive values.

40603: Argument error
Argument %s has a
not allowed negative value.
Check:
Set argument %s to positive.

40604: Argument error
Argument Tool has undefined
load of the tool.
Check:
Define the actual load of the tool
before use of the tool for jogging
or program movement.

40605: Argument error
Argument Tool has negative
load of the tool.
Check:
Define the correct load of the tool
before use of the tool for jogging
or program movement.

40606: Argument error
Argument Tool has at least
one inertia data component
with negative value.
Check:
Define all inertia data components
(ix, iy or iz) to actual
positive values.

40607: Execution error
Not allowed to change run mode
from forward to backward or vice versa
during running a circular movement.
Check:
If possible, select the original run
mode and press start to continue the
stopped circular movement. If not
possible, move robot and program pointer
for a new start.

40608: Argument error
Orientation definition error
in %s.
Check:
All used orientations must be normalized
i.e. the sum of the quaternion elements
squares must equal 1.

40609: Argument error
Argument WObj specifies a mechanical unit with too long name.
Check:
Use max. 16 characters to specify the name of a mechanical coordinated unit.

40610: Argument error
Argument WObj specifies a mechanical unit name, which is not activated or unknown in the system.
Check:
The mechanical unit name defined in WObj must correspond to the name earlier defined in the system parameters and must be activated.

40611: Execution error
Not allowed to step backwards with this move instruction.
Check:
Step backwards to a position defined with another tool or work object could result in fualty path.

40612: Argument error
No argument programmed for the name of the output signal.
Check:
Possible to set one position fix IO such as digital, group of digitals or analog output signal during the robot movement.

40613: Argument error
Optional argument %s
can only be combined with output signal argument %s.

40614: Argument error
Argument %s
is not 0 or 1.
Check:
Digital output signals can only be set to 0 or 1.

40615: Argument error
Argument %s
is not an integer value.
Check:
Digital group of output signals, process identity or process selector can only have an integer value.

40616: Argument error
Argument %s
is outside allowed limits.
Check:
Used group of digital output signals can only be set within 0 to %s according configuration in the system parameters.

40617: Argument error
Argument %s
is outside allowed limits.
Check:
Used analog output signals can only be set within %s to %s according configuration in the system parameters.

40618: Argument error
Argument %s
contains an illegal interrupt number.
Check:
Input interrupt number is illegal because it has not been allocated by the instruction CONNECT.
CONNECT do allocation and connection of interrupt number to trap routine.

40619: Argument error
Argument %s
contains an interrupt number, which is already in use for other purpose.
Check:
Before reuse of an interrupt variable again in the program, cancel old interrupt generation and interrupt number with instruction IDelete.

40622: Argument error
The value of argument Time in ITimer is too low for cyclic interrupts.

40623: Argument error
The value of argument Time in ITimer is too low for single interrupts.

40631: Instruction error
Too many move instructions in sequence with concurrent RAPID program execution.
Check:
Edit the program to max. 5 MoveX Conc
in sequence on the basic execution level of the program.

40632: Instruction error
No move instructions with concurrent RAPID program execution are allowed within the StorePath-RestoPath part of the program.
Check:
Edit the program so it does not contain any MoveX Conc instructions within the StorePath-RestoPath part of the program.

40633: Reference error
Trigg parameter no %s reference to undefined trigg data.
Check:
Define trigg data by executing instruction TriggIO, TriggInt, TriggEquip or TriggSpeed before TriggL, TriggC or TriggJ.

40634: Reference error
Signal reference in parameter %s contains unknown signal for the robot.
Check:
All signals should be defined in the system parameters and should not be defined in the RAPID program.

40635: Reference error
Argument reference
in parameter %.16s
is not a entire persistent variable.
Check:
Not possible to use record component or array element in arg. %.16s.
Only possible to use entire persistent variables for Tool, WObj or Load in any motion instructions.

40636: Sensor error
No measurement from sensor.
Check:
Requested data is not available.

40637: Sensor error
Not ready yet.
Check:
Requested function is not ready yet.

40638: Sensor error
General error.
Check:
General error has occurred which is not specifically connected to the requested action. Read the block "Error log" if the function is available.

40639: Sensor error
Sensor busy, try later.
Check:
The sensor is busy with an other function.

40640: Sensor error
Unknown command.
Check:
The function requested from the sensor is unknown.

40641: Sensor error
Illegal variable or block number.
Check:
Requested variable or block is not defined in the sensor.

40642: Sensor error
External alarm.
Check:
Alarm from external equipment.

40643: Sensor error
Camera alarm.
Check:
Some error has been detected in the camera. Run Camcheck to test if the camera is OK.

40644: Sensor error
Temperature alarm.
Check:
The camera is overheated it needs more cooling air or water.

40645: Sensor error
Value out of range.
Check:
The value of the data sent to the sensor is out of range.

40646: Sensor error
Camera check failed.
Check:
The CAMCHECK function failed. The camera is broken. Send it for repair.

40647: Sensor error
Communication time out.
Check:
Increase the time out time and check the connections to the sensor.

40648: Search error
Not possible to do StorePath while searching on basic path level.
Check:
If using program with robot movement in TRAP, then such interrupt must be deactivated during any searching.
E.g. ISleep - SearchL - IWatch

40649: Path limit error
%s already done.
Check:
Instruction %s must first be executed, before a new %s can be done.

40650: Wrong param combination
Optional parameters and switches are not used in a correct combination.
Check:
No optional parameters and no switch keeps the old coordinate system.
The switch Old has the same function.
RefPos or RefNum has to be defined with Short, Fwd or Bwd.

40651: Use numeric input
Use numeric input for the position instead of a robtarget.
Check:
The position can not be defined with a robtarget for robot axes.
Use the optional parameter for numeric input of the position.

40652: Axis is moving
A Robot axis, an external axis or an independent axis is moving.
Check:
All Robot axes, external axes and independent axes have to stand still.
E.g Use MoveL with Fine argument for the Robot and ext. axes. And IndRMove for the independent axes.

40653: Switch is missing
One of the switch parameters %s or %s has to be defined.

40654: Axis is not active
The axis is not active or it is not defined.
Check:
The mechanical unit has to be activated and the axis has to be defined, before this instruction is executed and before a robtarget is saved.

40655: Axis is not independent
The axis is not in independent mode.
Check:
It is only possible to get the status from an axis in independent mode.

40656: Execution error
Not possible to set a new scale value for the AO signal.
Check:
The internal process, that control the output of the AO signal, are for some unknown reason "dead

40657: Execution error
The output of the AO signal are not TCP-speed proportional any more.
The reason could be following:
Check:
- Use of MoveX between TriggX instr.
- No setup of TriggSpeed in the used TriggX instr.
- The used ScaleLag is too small in relation to actual robot servo lag.

40658: Parameter error
Parameter %s can only be used, if parameter %s is greater than zero.
Check:
Parameter %s has effect only in the first TriggX, in a sequence of several TriggX, that controls the speed proportional AO signal.

40659: Undefined load
WARNING: Argument %.16s has undefined load (mass equal to 0 kg).
IMPORTANT TO DEFINE CORRECT LOAD to
avoid mechanical damages of the robot and to get good motion performance.
Check:
Define the actual load for the tool or the grip load before program movement or jogging.

40660: Undefined load
WARNING: Argument %.16s has undefined load centre of gravity.
IMPORTANT TO DEFINE CORRECT LOAD to
avoid mechanical damages of the robot and to get good motion performance.
Check:
Define the actual centre of gravity for the tool load or the grip load before program movement or jogging (cog.x, cog.y and cog.z can not be 0 mm at the same time

40661: Search error
The signal %.16s for the SearchX instruction is already high at the start of searching.

40662: Invalid worldzone type
The switch %.16s must be associated with a %.16s worldzone.
Check:
If use of switch Temp, the datatype must be wztemporary in WorldZone.
If use of switch Stat, the datatype must be wzstationary in WorldZone.

40663: World zone not in use
The '%.16s' argument of the instruction %.16s must refer to a worldzone that has been defined and activated by a WZLimSup or WZDOSet instruction.

40664: World zone already in use
The '%.16s' worldzone has already been defined and activated.
A world zone can only be defined once.
Check:
Use a worldzone with another name.

40665: Too many world zones
It is not possible to add the world zone %.16s.
The world zone table is full
Check:

40666: Illegal world zones
Worldzone ’%.16s’ is
defined locally in current routine
or in current module
Check:
Only a global entire world zone
argument can be used.

40667: Illegal world zones
Task %.16s:
WorldZone %.16s
is not entire data reference.
Check:
Only a global entire world zone
argument can be used.

40668: shapedata not in use
The ’%.16s’ argument of the
instruction %.16s must refer
to a defined shapedata.
Check:
A shapedata is used to store a volume
definition. It must have been defined
by WZBoxDef, WZSphDef or WZCylDef prior
to be used by WZLimSup or WZDOSet.

40669: Invalid volume definition
The shapedata defined by the
instruction %.16s doesn’t
define a valid volume.
Check:
Check previous volume definition
instruction.

40670: Invalid world zone
The index of the world zone argument
%.16s in %.16s
is not a valid index defined by
WZLimSup or WZDOSet.

40671: Illegal use of world zone
Task %.16s:
’%.16s’
argument for %.16s
must be a temporay world zone.

40672: World zone already in use
It is not possible to add the world
zone %.16s. Another world
zone with the same name is already
defined in the system.

40673: I/O access error
The signal given in parameter
%s is write protected
for RAPID access.
Check:
Select other user signal or
change the access mode for the signal

40674: I/O access error
The signal given in parameter
%s is not write protected
for user access from TP or RAPID.
Check:
Change the access mode to system type
for the signal in the I/O
configuration.

40700: Syntax error
Task %s: Syntax error
%s

40701: Program memory full
The task %s, has only
%i free bytes in its
user space
Check:
Remove some other module and try
again.

40702: File not found
%.40s
The file path or the file name is wrong
or the file doesn’t exist.

40703: Load error
%.40s
The program module couldn’t be loaded.
Check:
The program module have some errors.

40704: UnLoad error
%.40s
The program module couldn't be
unloaded
The reason could be:
- Module not loaded with Load instr.
- Not same file path as used for Load
Check:
The program module must have been loaded with the instruction Load.
The file path and name must be the same in the UnLoad and Load instruction.

40705: Syntax error
Task %s: Syntax error
%s
Check:
More syntax errors will follow this

40706: Loaded error
The program module is already loaded
Check:
The module name in the head of the file
%.40s
already exists in the program memory

40707: I/O Unit name invalid
Task %s:
The unit name
%.40s
doesn't exist or can't be disabled.
Check:
The unit name is misspelled, not defined or can't be disabled.

40708: I/O unit is not enabled
Task %s:
I/O unit
%.40s
was not enabled.
Check:
The maximum period of waiting time was too short. Increase the waiting time or make a retry.

40709: I/O unit is not disabled
Task %s:
I/O unit
%.40s
was not disabled.
Check:
The maximum period of waiting time was too short. Increase the waiting time or make a retry.

40710: Argument error
Task %.16s: Argument error
from %.16s !
The argument is expression value, is not present or is of the type switch.
Check:
Change the parameter
(%.16s) to a
valid one.

40711: Alias type error
Task %s:
The data types for the arguments FromSignal and ToSignal must be the same and must be one of signalxx. (signalai/ao, signaldi/do, signalgi/go)
Check:
Change the type
to a valid one.

40712: Event routine error
Task %s: Too many event routines the routine %s, will not be executed. Maximum 4 routines could be specified for each event
Check:
Encapslulate the routine in one of the others that are specified for the save event.

40713: Alias define error
Task %s:
FromSignal must be defined in the io-system. ToSignal must not be defined in the io-system, it should just be defined as a variable.

40714: Argument error
Orientation definition error
in %s.
Check:
This is probably an off-line generated "dummy" position (undefined orientation), that needs to be MODPOS.

40715: IOEnable call interrupted
Task %s:
Call to I/O unit
%.40s
was inerrupted.
Check:
IOEnable or IODisable have been exec from an other task. Make a retry.

40720: Alias IO installation
The task %s, could not
refresh all alias IO

40721: IO installation
The task %s, could not
refresh all IO
as RAPID symbols
Check:
Check the IO configuration.

40722: Mechanical units
The task %s, could not
refresh all mechanical
units as RAPID symbols
Check:
Check the Motion
configuration.

40900: Discr. App. System Error
%s

40901: Discr. App. Sys Message
WARNING: A new application or process
has been ordered
A warm start is required to install a new configuration

40902: Discrete Application
Task %s:
Error from %.16s. Can’t
find the process %.16s.
Check:
Check the installed process(es

5 Motion error messages

50001: Serious motion error
Not possible to proceed motion control
Check:
Start up the system again

50021: Joint position error
Actual position of joint %s
is too far away from the ordered position
Check:
Check trim parameters, external forces or hardware.

50022: Too low DC-link voltage
Check:
Check voltage from Motor On contactors
Replace DC-link

50023: Stop-/Restart error
The stop was made when too many move instructions were queued for execution.
Restart is not possible
Check:
Check the number of move instructions with concurrency. Move the start point and start a new movement.

50024: Corner path failure
A corner path was executed as stop point due to some of the following reasons:
1 Time delay
2 Closely programmed points
3 System requires high CPU-load
Check:
1 Reduce the number of instructions between consecutive move instructions
2 Reduce speed, use wider spaced points, use /CONC option
3 Increase ipol_prefetch_time

50025: Restart too far from path
Check:
Move back to path.

50026: Singularity or Zone error
1 Robot too close to singularity
2 MoveL to MoveJ corner zone error
Check:
1 Use the joystick to move away from the singularity or run a program in joint coordinates
2 Use fine point or modify position

50027: Joint Out of Range
Joint %s is out of working range
Check:
Use the joystick to move the joint into its working range

50028: Jog in wrong direction
Joint %s is out of working range
Check:
Use the joystick to move the joint in opposite direction.

50029: Robot outside its limits
The robot has reached the configuration limit for the parallelogram transmission.
Check:
Use the joystick to move the involved joint into the working range again.

50030: Robot outside its limits
Jogging was made in wrong direction when
parallelogram was out of working range
Check:
Use the joystick to move the joint in opposite direction.

50031: Command not allowed.
System parameters cannot be changed in MOTORS ON state.
Check:
Change to MOTORS OFF.

50032: Calibration command error
An attempt was made to calibrate while in MOTORS ON state.
Check:
Change to MOTORS OFF.

50033: Commutation command error
An attempt was made to commutate the motors in MOTORS ON state.
Check:
Change to MOTORS OFF.

50035: Synchronization error
An attempt was made to synchronize in MOTORS ON state.
Check:
Change to MOTORS OFF.

50036: Correct regain impossible
Correct regain impossible. A stop occurred with too many close points with corner zones. At restart the robot will move to a point farther forward in the program.
Check:
Reduce the number of close points, increase the distance between them or reduce the speed.

50037: MOTORS ON order ignored
MOTORS ON order ignored since the previous stop was not yet acknowledged.
Check:
Order MOTORS ON again.

50041: Robot in a singularity
The Robot is too close to a singularity.
Check:
During program execution, use SingArea instruction or joint interpolation.
During jogging, use axis by axis.

50042: System error
Check:
Increase the distance between close points and/or decrease speed and/or change acceleration value.

50050: Position outside reach
Position for IRB joint %.f
is outside working area.
Check:
Check the work object.
Check the joint working range.
Move the joint in joint coordinates.

50052: Joint speed error
The speed of joint %s is too high
relative the ordered speed
Check:
1. Check the tune parameters, external forces on the joint and hardware.
2. Reduce programmed speed.

50053: Revolution counter error
Too big difference between the counter in the serial measurement board and the expected value in the robot computer for joint %s
Check:
Update the revolution counter
Replace serial measurement board

50055: Joint load error
Actual torque on joint %s too high
May be caused by incorrect load data, too high acceleration, high external process forces, low temperature or hardware error
Check:
1. Check load data
2. Reduce acceleration or speed
3. Check hardware

50056: Joint collision error
Actual torque on joint %s is
higher than ordered while at low
or zero speed.
Check:
May be caused by jam error (the arm has got stuck) or hardware error.

50057: Joint sync. error
The position of joint %s after
power down/failure is too far away
from the position before the power
down/failure.
Check:
Make a new update of the revolution counter.

50058: Tool coord. sys. error
The z-direction of the tool coordinate system is almost parallel with the path direction.
Check:
Change the tool coordinate system to achieve at least 3 degrees deviation between z-direction and path direction.

50059: Frame error
The definition of robot
fixed tool is not correct.
Check:
Check the tool and object data.

50060: Frame error
The definition of robot
fixed tool is not correct.
Check:
Check the tool and object data.

50061: Frame error
The definition of robot
fixed tool is not correct.
Check:
Check the tool and object data.

50062: Circle programming error
Start and end positions for the circle are too close.

50063: Circle programming error
The circle position is too close to the start or end position of the circle.

50065: Kinematics error
The destination of the movement is outside the reach of the robot or too close to a singularity.
Check:
Change the destination position.

50066: Robot not active
Attempt to coordinate motion or
calculate position of deactivated
robot %s.
Check:
Activate robot via the Motion Unit key, then Jogging window, or program.
Check work object and program.

50067: Unit not active
Attempt to coordinate motion or calculate position of deactivated single unit %s.
Check:
Activate unit via Motion Unit key, then Jogging window, or program. Check work object and program.

50076: Orientation def. error
Orientation is incorrectly defined.
Check:
Make an accurate normalization of the quaternion elements.

50078: Too many close positions
Too many consecutive closely spaced positions.
Check:
Increase the distance between consecutive close positions.

50079: Wrist weaving
not possible.
Check:
Use smaller weaving amplitude or a larger TCP.

50080: Position not compatible.
Position cannot be reached with the given robot configuration
Check:
Modify the robot position in the program.

50082: Deceleration limit
Calculation of joint deceleration time exceeds internal limits for this motion.
Check:
You cannot proceed without removing the cause(s) of this error (see Check).
Increase path resolution (sys param or by PathResol for critical movements).
Reduce speed, use fine, increase AccSet, increase Queue time, avoid singularity (SingAreaWrist), inc. dynamic resol.

50083: Speed lowered by system.
The speed has been lowered by the system due to dynamic limitations.
Check:
Decrease speed and/or do not use close positions at high speed and/or increase acceleration (if below 100%

50085: Too many user frames.
For mech_unit %s more than one user frame has been defined.
Check:
Take away one user frame or define one more mech_unit.

50086: Singularity calc. error
Too close to wrist singularity with respect to numerical resolution for joint 4 of IRB.
Check:
Change destination position a few increments.

50087: Singularity problems.
Too close to wrist singularity with respect to numerical resolution for joint 6 of IRB.
Check:
Change destination position a few increments.

50088: Restart not possible.
It is not possible to restart the path due to a previous error.
Check:
Move the program start point and start a new movement.

50089: Weaving changed
The ordered weaving is not achieved due to: high weaving frequency, not allowed shift of weave method or that SingArea/Wrist is used with wrist weave
Check:
Increase weave length or period time.
Don’t shift between arm and wrist weave.
Use SingArea/Off with wrist weave.

50091: Restart not possible.
Restart no longer possible. Change of unit state made restart of program impossible.
Check:
Move the program pointer and start a new movement.

50092: Axis computer comm. error
Incorrect response from axis computer
Check:
Check motion configuration parameters.
Check axis computer hardware.

50094: ServoTune not possible.
Tuning is not implemented for the specified Joint.

50095: Cannot access joint.
Cannot access external joint. Check configuration and activation of external Joints.

50096: ServoTune not allowed.
Tuning is not allowed for the specified joint.

50100: Manipulator error
There are more configuration or numerical errors in motion domain.
Check:
Correct previous ones and try again.

50101: Manipulator config. error
'%s' is not free
for the param. '%s'
in type '%s' named
'%s'.
Check:
Use another one.
For internal names, see moc_chk.log.

50102: Manipulator config. error
'%s' used in the
parameter '%s' in
type '%s' named
'%s' is not
defined.
Check:
Use another one that is defined or define the used one.
For internal names, see moc_chk.log.

50103: Num. error in manipulator
The orientation defined by quaternions including '%s' in
the type '%s' named
'%s' is not
normalized.(SQRSUM =1)
Check:
Check the quaternions and/or recalculate them.
For internal names, see moc_chk.log.

50104: Num. error in manipulator
The parameter '%s'
in type '%s' named
'%s' is not '%s'.
Check:
Check the value.
For internal names, see moc_chk.log.

50128: Manipulator error
Terminating the topic check for manipulator due to earlier errors.
Check:
Correct the reported errors and run topic check again.

50130: Synchronization failed.
Synchronization failed for joint
%s.
Check:
Make a new synchronization.
Restart System.

50131: Calibration failed.
Calibration failed for joint
%s.
Check:
Make a new calibration.
Restart System.

50132: Commutation failed.
Commutation failed for joint
%s.
Check:
Make a new commutation.
Restart System.

50133: Test signal error.
No test signals are available for the master robot.

50134: Corr. vector warning
Sensor correction vector calculations failed due to previous error.

50135: SoftAct not possible.
Soft servo is not possible to activate.

50137: Fine point inserted
Corner zone is changed to fine point
Too many consecutive Move instructions without fine point

50138: Arm check point outside
The robot has reached the limit for arm check point
Check:
Use the joystick to move the involved joint into the working range again

50139: Arm check point outside
Jogging was made in wrong direction when
arm check point was out of working range
Check:
Use the joystick to move the joint in opposite direction.

50140: Payload too large
Heavy payload caused static torque limit to be exceeded on joint %s
Check:
Check and reduce payload
for arm and/or wrist.
Reduce joint working range to decrease static torque due to gravity.

50141: Jog or Speed error
1. Jogging error
2. High speed error
3. Robot too close to singularity
Check:
1. Jog with incremental movement
2. Reduce the programmed speed

50142: Manipulator config. error
Configuration of the manipulator failed.
Check:
Check the parameter values under System parameters:Manipulator.

50143: Robot axes config. error
Actual configuration is not the same as ordered and/or reorientation of joint 4/6 is too large.
Check:
Use SingArea_Wrist, ConfL_Off, modify position or insert intermediary point.

50144: Displ frame uncertain.
Calibration of displ frame uncertain
1. Wrong TCP
2. Ref. points inaccurate
3. Ref. points badly spaced
Check:
If estimated error is unacceptable:
1. Verify that correct TCP is used.
2. Try more than 3 ref. points.
3. Be careful when positioning robot to ref. points.

50145: Kinematic limitation
Kinematic limitation, no solution found.
1. Long segment.
2. Position close to singularity.
3. Joint 1, 2 or 3 out of range.
4. Position outside reach.
Check:
1. Insert an intermediary point to reduce the length of the segment.
2. Use MoveAbsJ.
3-4. Check working range.

50146: Restart limitation
Corner path executed as a stop point.
Power fail restart not possible near the stop point.
Check:
Use finepoint in the Move-instr before RestoPath, ActUnit, Wait or Stop-instr to make power fail restart possible.

50147: Power fail restart failed
Re-creation of the path failed
Check:
Move the start point and start a new movement.

50151: MOC_MAILBOX1_ERROR
The axis computer driver failed
to generate a new mailbox 1 interrupt
since the previous interrupt has
not been serviced properly.
Check:
Reload system
Replace robot computer board

50152: MOC_MAILBOX2_ERROR
The axis computer driver failed
to generate a new mailbox 2 interrupt
since the previous interrupt has
not been serviced properly.
Check:
Reload system
Replace robot computer board

50153: Command not allowed
The given instruction, or command, was
not allowed since the robot program was
executing in a hold state.
(%s %d %d)
Check:
Modify program or stop program execu-
tion
before issuing command.

50154: Command not allowed
SingAreaWrist mode interpolation is
not supported for the IRB6400C robot.
Check:
Replace SINGAREAWRIST instruction
with
SINGAREAOFF.

50155: Power fail restart failed
Not possible to restart the Move-instr
before RestoPath, ActUnit, Wait or
Stop-instr
Check:
Make program free from MOTION
WARNING
50146 Restart limitation, by changing
the Move-instr to finepoint
Move the start point and start
a new movement.

50156: Independent joint error
Joint %s is not configurated as an
independent joint.
Check:
Modify the program or configurate
the joint as an independent joint.

50157: Corr. vector warning
Sensor correction vector X calculations
failed due to previous error.

50158: Sensor process missing
Sensor process missing during
initialization.
Named sensor process %s
could not be found or initialized.
Check:
Check process name in motion and
process configuration files.

50159: No external process
Attempt to coordinate motion or
calculate position of single %s
without an external process.
Check:
Check process name in motion and
process configuration files.

50160: Cannot reach position
Programmed position of indep. joint
%s is outside working range and
thus cannot be reached.
Check:
Change the position.
Check the joint working area limits.
Check the used work object.

50161: Singularity area
Robot is close to a singularity.
Work area with kinematic limitations.
Check:
During jogging, use axis by axis.
During program execution,
use MoveAbsJ.

50162: Internal position error
Error caused by internal
numerical limitation
Check:
Reset independant joint
Reduce work area if extended
Remove or separate close points

50163: External Pos adjustment
External Pos adjustment too large.
TCP speed, orientation speed, or
external position speed exceed
allowed robot performance.
Check:
1. Reduce programmed TCP and
   orientation speeds
2. Modify the path
3. WaitWObj closer to sync
4. Run in AUTO

50164: Ind. deactivation error
Independent deactivation error
Deactivation of mechanical unit may not
be done while in independent mode.

50167: Warning: new sync
Warning: a new object sync signal has
arrived while conveyor is active and
program is running.

50168: New object sync
New object sync arrived while conveyor
was tracking the previous object.
Cannot track two objects simultaneously
Check:
Reduce speed of conveyor
Increase programmed speed

50170: Process missing
External control process missing during
initialization.
Named process %s
could not be found or initialized.
Check:
Check process name in motion and
process configuration files.

50171: Speed too low
Numerical problem when interpolation
of long segments with low speed and
heavy external axes or
when interpolation close to singularity
Check:
Split segments with long interpolation
time (path_resolution * 4 minutes) or
change to joint interpolation or move
position away from singularity.

50172: MoveJ not allowed
MoveJ not allowed with work object
coordinated with external
position mechanical unit.
Check:
Change interpolation mode or
work object.

50173: Use fine point
Use fine point when changing tool
or work object coordination
when work object is coordinated with
external pos mechanical unit.
Check:
Create a fine point and
then change the tool.

50174: WObj not connected
The WObj is not connected to the
moving conveyor. Robot TCP cannot be
coordinated to work object.
Check:
Check for missing WaitWObj.
Check for DropWObj occuring before end
of coordination.

50175: Conveyor moving
Conveyor moving while attempt to
coordinate robot TCP to conveyor work
object while in prohibited mode.
Check:
It is not possible to coordinate to
conveyor while in Manual Reduced
Speed,
or stepping in Auto, and the conveyor
is moving.

50176: Conveyor not active
Conveyor mechanical unit was not active when attempt to coordinate robot TCP to conveyor work object.
Check:
Make sure conveyor mechanical unit is active. Check for fine point for last coordinated motion before DeactUnit.

50177: Unable to restart
Conveyor moving while attempting to restart or before pressing
Stop or stepping through program
Check:
Make sure conveyor is standing still.
Move the program pointer.

50178: Non optimal movement
Non optimal movement
Required torque too high
Manual adjustment of acceleration or speed is needed.
Check:
Reduce acceleration (AccSet 50 100) in this movement, restore it afterwards (AccSet 100 100). Optimize performance
by search for max acceleration 50-99
Alternatively, reduce speed.

50180: Corr. vector warning
Sensor correction vector calculations failed due to previous error.

50181: Out of coupled range
Joint %s and %s are out of coupled working range.
Check:
Use the joystick to move joints into their coupled working range.

50182: Jog in wrong direction
Joint %s and %s are out of coupled working range.
Check:
Use the joystick to move joints into their coupled working range.

50183: Robot outside work area.
The robot has reached the
World Zone %s.
Check:
Check the reason of the World Zone.
Use the joystick to move the robot out of the World Zone if needed.

50184: Corr. vector warning
Sensor correction vector calculations failed due to previous error.

50185: Corr. vector warning
Sensor correction vector calculations failed due to previous error.

50186: Coordination error.
Not possible to run robot %s
with coordinated base frame.
Function not installed in this system
Check:
Install the option Advanced Motion.

50187: Coordination error.
Not possible to coordinate user with robot %s
Function not installed in this system
Check:
Install the option Advanced Motion.

50188: Non optimal movement
Non optimal movement
Required torque too high
Manual adjustment of weave frequency or amplitude is needed.
Check:
Reduce weave frequency or
weave amplitude in
this movement
Alternatively, reduce speed.

50189: Relay signal not found
The signal %s for relay %s
is not found in the I/O configuration.
Check:
The mechanical unit using this relay is ignored.
Check I/O signal definitions and System Parameters definition of Manipulator, Types: Relay.

50190: Permanent ipol lock error
Scanned no of active joints not equal expected no of joints.
Check:
Check configuration of unit using general kinematics.

50191: Too many TCP speed's
The number of TCP speedsinone' segment is too large. Maximum number of TCP speed's is %d.
Check:
Check if one segment has too many TCP speed's set or if a sequence of segments have increasing DipLag.

50192: Jogging error
Jogging is started too soon after program stop.
Check:
Restart and try again.

50193: Joint sync. error
The speed of joint %s before power down/failure was too high.
Check:
Make a new update of the revolution counter.

50194: Internal position error.
Error caused by internal numerical limitation.
Log. joint number %2.0f.
Check:
Dyn step 0 = %f
Dyn step 1 = %f
Dyn step 2 = %f
Axc step =

50195: Independent move error
Joint %s cannot be moved in independent mode.

50196: Calibration error.
Points 0 and 1 too close

50197: Calibration error.
Points 0, 1, 2 on a line or point 2 too close to points 0 or 1
Check:

50198: Calibration error.
Internal error
Check:
Report the occurance.

50199: Calibration error.
External joints have been moved during calibration.
Check:
Avoid moving external joints.

50200: Torque error.
Torque error due to high speed.
Check:
1 Check load data.
2 Reduce speed.

50201: Orientation outside reach
The error of the programmed orientation exceeds the acceptance limit.
Check:
1 Adjust robtarget orientation.
2 Adjust/check orientations of currently used frames: tool frame, base frame, user frame, object frame.

50202: No dc link configurated
The dc-link(%s) and drive unit(%s) used by joint: %s is not connected to same serial link.
Check:
Select another dc-link.
Change serial link for drive unit.
Set the parameter '-no_dc_link' to TRUE.

50203: Measurement node used
The measurement node for joint %s is already used.
Check:
Select another node.

50204: Motion supervision
Motion supervision triggered
for joint %s.
Possible causes: collision,
incorrect load definition,
external process forces.
Check:
If load definition incorrect,
use load identification.
If external forces, use RAPID
command or system parameters
to raise supervision level.

50205: Data logger error:
%s
Check:
Solution:

50206: Probe warning
Probe buffer is full.

50207: Add intermediate point
Add intermediate point
not coordinated to external pos
mechanical unit.
when changing conveyor.
Check:
Create an intermediate point
then change the conveyor.

50208: Missing function
Friction Compensation can not be
activated for joint %s.
Check:
Install the option Advanced Motion.

50209: Kinematic limitation.
No accepted solution found.
Residual: %ld deg in orientation,
%ld mm in x,
%ld mm in y,
%ld mm in z.
Check:
Insert an intermediary point.
Check singularity.
Increase position and orient. tolerance.
Use MoveAbsJ.
Check working range.

50210: Load identification error.
Check:
Possible problem:
Load identification not allowed
for this robot.
Configuraion angle is too small.

50211: External Drive error.
External controlled drives can not
be used without the option
'External Drive'

50212: General kinematics
General kinematics can not
be used without the option
'General Kinematics'

50213: Frame rotation error.
Possibly due to unnormalized
quaternion
or position out of bounds.
Check:
Check the normalization of
input quaternions
or magnitude of input positions.

50214: Configuration error of work area.
Possibly the defined work area
are larger than max allowed area.

50215: Load identification error.
Axis %d will move outside
working range.
Check:
Move the axis to a position further from the
working range
limit.

6 Operator error messages

60001: %s missing.
Tool %s is not used in
current program.
Maybe because it has been deleted or
it is not defined.
Check:
Change to another tool using the
Jogging window.

60002: %s missing.
Wobj %s is not used in
current program.
Maybe because it has been deleted or
it is not defined.
Check:
Change to another workobject using
the Jogging window.

60003: Directory not created!
The directory %s cannot be created.
Probably, because directory already
exists or the disk is write-protected.
Check:
Check if directory exists or if disk
is write-protected.
Check also if space on disk is enough.

60004: Robot Hold confusion!
The used tool and the used work object
cannot both, at the same time, be held
by robot or be stationary.
Check:
Check the robhold component of the
used tool and work object.

60005: %s missing!
The workobject %s contains
a coordinated mechanical unit which
cannot be found.
Check:
Check the mechanical unit component of
the workobject.

60006: %s Userframe!
The workobject %s contains
a coordinated mechanical unit which
has no defined userframe.
Check:
Check the mechanical unit component of
the workobject.

60007: Jogging not permitted!
Jogging cannot be done in this mode.
Check:
Release the joystick and enabling
device and repeat.
Check also active mechanical unit.

60008: Tool mass undefined!
Jogging cannot be done if the used tool
has an undefined mass
Check:
Enter a value for the mass, into the
tooldata for the used tool.

60009: Unsynchronized robot!
The robot or external axis are
unsynchronized.
Check:
Synchronize robot or external axis.

60010: Orientation error!
Orientation in %s is unnormalized.
Check:
Check orientation value.

60011: Parameter faults!
Loading of parameters in
%s
cannot be fulfilled.
For reason, see
%s
Check:
Copy the file
%s
to a floppy and examine reasons
using an ordinary text editor!

60012: No Parameters loaded!
There are no parameters in
%s
Check:
Check the file
%s
using an ordinary text editor!

60013: Jogging not permitted!
Jogging of mechanical unit is
not possible.
Unit is not activated.
Check:
Activate the mechanical unit.

60014: Disk is full!
No info is saved in Change Log about
the parameter change because no space
available on disk.
Check:
Try to delete files or
reorganize your disk.

60015: PP cannot be set!
PP cannot be set to routine
'%s' because it has parameters.
Check:
Make a routine which call '%s' or
remove the parameters.

60016: PP cannot be set!
PP cannot be set to routine
'%s' because it resides in a
module which has NOSTEPIN as
module attribute.
Check:
Copy the routine '%s' to
another module or change the
module attribute.

60017: PGM_TELLBACK code %d
Check:
No more information available.

60018: RAPID syntax error!
The program cannot be loaded because
of
syntactical error(s).
Check:
A RAPID syntax check program for the
PC
or QuickTeach can be used to detect the
error(s). The file PGMCPL1.LOG on the
internal RAM disk contains information
about the error(s).

60019: Data input error!
The component '%s' in data
type '%s' is not correct.
The limits are
%s!
Check:
Check data and enter the correct
value.

60020: PP cannot be set!
PP cannot be set to routine
'%s' because it is defined as
a trap routine.
Check:
Change the definition for the
routine '%s'
to 'Procedure'.

60021: Cannot show items!
The number of selected items exceeds the
current memory limit specified for this
configuration. The items can thus not
be shown.
Check:
Reduce the number of data or
change the configuration to a memory
board with more memory.

60022: Cannot show all items!
Only %d variables (out of %d)
will be listed.
All variables cannot be shown because
the current memory limit specified for
this configuration will be exceeded.
Check:
Reduce the number of data or
change the configuration to a memory
board with more memory.

60023: Limit ModPos!
You cannot modify this position because
limit modpos is activated with ABS.
ABS, absolute mode, means that the
original position should be saved.
This cannot be done while tuning.
Check:
If executing, stop the program. Modify
the position in the Program Window.
This will create an original position.
This position will thereafter allow
tuning. Limits are set by Limit Modpos.

60024: Outside Limits!
The change is either outside the
internal limit 10 mm or exceeds the
limit set by limit modpos parameter
Max Trans.
Check:
A single change cannot exceed 10 mm.
Do the change in smaller steps.
If Limit Modpos is set and the parameter
Max Trans is less than 10 mm this
parameter has to be changed.

60025: Name not allowed!
The name already exist or is a reserved word.
Check:
Please use an other name. See list of reserved words in Rapid manual.

60026: Program memory soon full
Save program or take other appropriate actions.

60027: Cannot calc. tool frame!
It is not possible to calculate the tool frame with the selected approach points.
Check:
Select new approach points as accurate as possible.

60028: Cannot show all signals!
Only %d signals (out of %d) will be listed.

60029: Cannot show all units!
Only %d units (out of %d) will be listed.

60030: Too many signals!
Too many signals are selected for the Most Common list. Only the first %d will be listed.

60031: Incompatible file!
The version of the configuration file is not compatible with this system.
Check:
Check the version of the configuration file.

60032: Cannot calc. user frame!
It is not possible to calculate a user frame with the selected approach points.
Check:
Select new approach points as accurate as possible.

60033: Cannot calc. object frame!
It is not possible to calculate an object frame with the selected approach points.
Check:
Select new approach points as accurate as possible.

60034: Volume is not aviable !

60035: Can not save disk is full !

60036: File not found !

60037: Cant open/create file !
No file descriptor aviable

60038: Invalid number of bytes !

60039: File already exists

60040: Illegal name !

60041: Can not delete root !

60042: Not file !

60043: Not directory !

60044: Not same volume !

60045: File/directory is read only !

60046: Root directory if full !

60047: Directory is not empty !

60048: Bad disk !

60049: No lable !

60050: Invalid parameter !

60051: No contig space !

60052: Can not change root !

60053: File descriptor obsolete !

60054: Deleted !

60055: No block device !

60056: Bad seek !

60057: Internal error !

60058: File/directory write only !

60059: Disk changed !

60060: No disk in driver !

7 IO & Communication error messages

71000: Bus/Type incompatible
DescriptionReason:
- Unit %s has a type
  that isn't compatible with it's bus
Check:
1. Change the bus for the unit
2. or change the unit type

71001: Duplicated address
DescriptionReason:
- Same address for unit %s and %s
Check:
1.Check the address
2.Check the bus

71002: Invalid unit type
DescriptionReason:
- Unit %s: has an unspecified
  unit type named: %s.
Check:
1.Check the unit type against the one
  specified for the unit

71003: Invalid unit
DescriptionReason:
- The unit specified for the signal %s
  is not specified in the unit section
Check:
1.Change the name of the unit
2.Add a new unit to the unit list

71004: Invalid signal length
DescriptionReason:
- The length of the digital
  signal %s must be 1
Check:
1.Change the length to 1 or remove
  the statement.

71005: Filter time invalid
DescriptionReason:
- Signal %s: The passive filter time
  should be 0 or %d - %d ms
Check:
1.Change the filter time

71006: Filter time invalid
DescriptionReason:
- Signal %s: The active filter time
  should be 0 or %d - %d ms
Check:
1.Change the filter time

71007: Logic. value out of range
DescriptionReason:
- Signal %s: Logical Max is less or
  equal to Logical Min
Check:
1.Correct the values to be max greater
  than min

71008: Phys. value out of range
DescriptionReason:
- Signal %s: Physical Max is less or
  equal to Physical Min
Check:
1.Correct the values to be max greater
  than min

71009: Type invalid
DescriptionReason:
- Signal %s: the type of signal
  is invalid
Check:
1.Change the type

71010: Signal out of range
DescriptionReason:
- Signal %s: the physical signal number + length -1 cannot exceed max. physical signal number for the unit. Maximum value = %d
Check:
1.Change the physical signal number.
2.Change the length.
3.Check the signal type.

71015: Digital Input overflow
DescriptionReason:
- Number of digital input channels for board %s is greater than %d
Check:
1. Reduce the number digital inputs

71016: Digital Output overflow
DescriptionReason:
- Number of digital output channels for board %s is greater than %d
Check:
1. Reduce the number of digital outputs

71017: No activate signal
DescriptionReason:
- Missing activate signal for cross
Check:
1. One activate signal must be given

71018: Activate signal overflow
DescriptionReason:
-Number of activate signals for cross too high
Check:
1. Only one activate signal must be given

71019: Missing signal definition
DescriptionReason:
-The signal: %s, at cross is not defined
Check:
1. Define the signal name in signal section

71020: No result signal
DescriptionReason:
- Missing result signal
Check:
1. At least one result signal must be given

71021: Duplicate cross signals
DescriptionReason:
- The signal: %s, appears both as FROM and as TO.
Check:
1. The same signal can not be given for both FROM and TO

71022: Physical max too high
DescriptionReason:
- Signal: %s
- The physical max value > %.3f
Check:
1. Change value in configuration

71023: Physical min too low
DescriptionReason:
- Signal: %s
- The physical min value < %.3f
Check:
1. Change value in configuration

71024: Physical value too high
DescriptionReason:
- Signal: %s
- Current value = %.1f >
- Maxvalue = %.1f
- Value set to Maxvalue
Check:
1. Change physical max value in configuration

71025: Physical value too low
DescriptionReason:
- Signal: %s
- Current value = %.1f <
- Minvalue = %.1f
- Value set to Minvalue
Check:
1. Change physical min value in configuration

71026: Logical value too high
DescriptionReason:
- Signal: %s
- Current value = %.1f - Maxvalue = %.1f >
- Value set to Maxvalue
Check:
1. Change logical max value in configuration

71027: Logical value too low
DescriptionReason:
- Signal: %s
- Current value = %.1f <
- Minvalue = %.1f
- Value set to Minvalue
Check:
1. Change logical min value in configuration

71033: Dig. input out of range
DescriptionReason:
- The number of digital inputs is out of range at board address %d max inputs are %d
Check:
1. Change the configuration for the board

71034: Dig. output out of range
DescriptionReason:
- The number of digital outputs is out of range at board address %d max outputs are %d
Check:
1. Change the configuration for the board

71036: Name out of range
DescriptionReason:
- The number of characters in
- name %s
- is greater than %d characters or
- the name is missing.
Check:
1. Give a new name that fits within the limits.

71037: IO Cross connection fault
DescriptionReason:
- The signal %s appears on both FROM and TO in the same chain
Check:
1. Correct the configuration for the cross connections where the signal above is connected.

71038: IO Cross depth to high
DescriptionReason:
- The Cross connection in the same chain
- is too deep.
- First signal name: %s
Check:
1. Make the Cross connection less deep.

71041: Analog output overflow
DescriptionReason:
- Number of analog output for
- board, %s is greater than %d.
Check:
1. Reduce the number of analog outputs.

71042: Analog inputs overflow
DescriptionReason:
- Number of analog inputs for
- board, %s is greater than %d.
Check:
1. Reduce the number of analog inputs.

71043: Signal type error
DescriptionReason:
- The type specified for signal %s can’t be connected to specified board
Check:
1. Change to another type.
2. Change to another board.

71044: Physical signal overflow
DescriptionReason:
- The range of phsig, or length, or phsig and length for signal %s is greater than %d
Check:
1. Change the physical signal number
2. Change the length.

71045: Filter specification err.
DescriptionReason:
- Signal %s : No filter time can be specified for this type of signal.
Check:
1. Set filter time to 0 or remove the statement.

71046: Scaling error
DescriptionReason:
- Signal %s: No scaling can be done.
Check:
1. Remove the scaling statements.

71049: Parameter Invert error
DescriptionReason:
- Signal %s: This type of signal can't be inverted.
Check:
1. Only digital signals can be inverted.

71050: Cross signal not digital.
DescriptionReason:
- Signal %s: Is not a digital signal
Check:
1. Only digital signals can be cross. connected.

71052: Cross table full.
DescriptionReason:
- The sum of different FROM signals added with total sum of TO signals must not exceed %d
Check:
1. Reduce the number of signals.

71053: Connection to board down
DescriptionReason:
- Can't access the board due to communication is down
Check:
1. Check the communication cable to the board
2. Check if the board is switched off

71054: Wrong signal type
DescriptionReason:
- Signal %s:
The type of signal is wrong
Check:
1. Change the type

71055: Invalid signal name
DescriptionReason:
- Symbol %s: is not defined
Check:
1. Change the symbol name above

71056: Power fail restore full
DescriptionReason:
- Symbol %s: could not be setup for power failure restore.
The table for power fail is full.
Check:
1. Increase the table size in startup file.
2. Remove some other signal from restore list.

71058: No contact with I/O unit
DescriptionReason:
- No contact with I/O unit: %s on bus: %s
Check:
1. Check the addresses on all I/O units connected to the bus
2. Change the address

71059: Error config. Can node
DescriptionReason:
- Error when configuring Can node connected at node address %d
Check:
1. Correct the configuration for the Can node at given address.

71061: I/O bus error
DescriptionReason:
- An abnormal rate of errors on bus %s has been detected.
Check:
1. Check the bus.
2. Restart System.

71072: No save set on signal
DescriptionReason:
- Signal %s
  has not Set the Store attribute to YES
Check:
1. Set the Store attribute.

71073: Error on I/O Bus
DescriptionReason:
- An abnormal rate of errors on the
  %s Bus has been detected.
  The connector for External CAN I/O
  has been switched off
Check:
1. Check bus terminators.
2. Check I/O bus for short-circuit.
3. Restart system.

71074: Config. out of range
DescriptionReason:
- The IBS starting quarter and rack
  size is out of range for
  board %s
Check:
1. For starting quarter 1 rack size
   must be less than 5

71076: Comm error from rtp1
DescriptionReason:
- No response from the
  serial line
Check:
1. Check the device or connection

71077: Comm error from rtp1
DescriptionReason:
- Not possible to deliver
  the received message
Check:
1. Check the communication flow

71078: Comm error from rtp1
DescriptionReason:
- The response from the device
  has a non valid frame sequence
Check:
1. Check for noise on the serial
   line

71079: Pulsing group output
DescriptionReason:
- Pulsing group output not allowed.
Check:

71080: Unit type table full.
DescriptionReason:
- The number of unit types
  must not exceed %d
Check:
1. Reduce the number of unit types.

71081: Physical table full.
DescriptionReason:
- The number of physical signals
  must not exceed %d
Check:
1. Reduce the number of physical
   signals.

71082: Signal table full.
DescriptionReason:
- The number of user defined signals
  plus panel signals must not
  exceed %d
Check:
1. Reduce the number of signals.

71083: Symbol table full.
DescriptionReason:
- The number of symbols
  must not exceed %d
Check:
1. Reduce the number of symbols.

71084: Triggr table full.
DescriptionReason:
- The number of Subcribed signals
  must not exceed %d
Check:
1. Reduce the number of Subcribed
   signals.

71085: Unit table full.
DescriptionReason:
- The number of boards
  must not exceed %d
Check:
1. Reduce the number of defined boards.

71090: Invalid unit type.
DescriptionReason:
- The vendor id read from
  unit %s doesn't match
  value in unit type configuration
  Configuation: %d Actual: %d
Check:
1. Change vendor id in configuration.
2. or check module.

71091: Invalid unit type.
DescriptionReason:
- The profile read from
  unit %s doesn't match
  value in unit type configuration.
  Configuration: %d Actual: %d
Check:
1. Change profile in configuration.
2. Check that the type of board is
   correct.

71092: Invalid unit type.
DescriptionReason:
- The product code read from
  unit %s doesn't match
  value in unit type configuration.
  Configuration: %d Actual: %d
Check:
1. Change product code in configuration.
2. or check module.

71093: Invalid unit type.
DescriptionReason:
- The major revision read from
  unit %s doesn't match
  value in unit type configuration.
  Configuration: %d Actual: %d
Check:
1. Change major revision.
2. or check module.

71094: Too many cross-actors def
DescriptionReason:
- The cross-connection has too
  many "From" signals : %s
Check:
1. Check the cross configuration.

71095: Too long cross-actor str
DescriptionReason:
- The "From" part in the cross-string
  is too long : %s
Check:
1. Check the cross configuration.

71097: Parameter store error
DescriptionReason:
- Signal %s: This type of signal
 can't have store option.
Check:
1. Only digital output signals can
   have store.

71098: NFS server lost
DescriptionReason:
- The contact with the NFS server
   '%s' was lost.
Check:
1. Check the NFS server.
2. Check the network connection.
3. Check the configuration.

71099: Trusted NFS server lost
DescriptionReason:
- The contact with the trusted
  NFS server '%s'
  was lost.
Check:
1. Check the NFS server.
2. Check the network connection.
3. Check the configuration.

71100: Bus table full.
- The number of buses
  must not exceed %d
Check:
1. Reduce the number of buses.
2. Increase the number of buses
   allowed.

71101: Unknown bus name.
DescriptionReason:
- Board %s: Unknown bus
  name %s
Check:
1. Change the bus name for the board
   at unit type configuration.

71102: DeviceNet incompatible
DescriptionReason:
- Node %d:
  Internal info:
  %s,%d
  (%s)
Check:
1. Disconnect the node from bus
2. Contact ABB.

71103: Error on I/O Bus
DescriptionReason:
- An abnormal rate of errors on the
  %s Bus has been detected.
  The connector for Robot CAN I/O
  has been switched off
Check:
1. Check bus terminators.
2. Check I/O bus for short-circuit.
3. Restart system.

71104: Error on I/O Bus
DescriptionReason:
- An abnormal rate of errors on the
  %s Bus has been detected.
The connectors for External CAN I/O and Robot CAN I/O has been switched off
Check:
1. Check bus terminators.
2. Check I/O bus for short-circuit.
3. Restart system.

71105: Disable group failed
DescriptionReason:
- Failed to disable unit %s
  at address %d.%d
Check:
1. Check I/O system parameters
2. Check unit

71106: dsqc344 board failure
DescriptionReason:
- Faulty or re-started IBS board
  Board internal error code: %d
Check:
1. Check above error, see IBS manual:
   Firmware Service and Error Messages
2. Update board firmware
3. Replace InterBus-S board

71107: InterBus-S bus failure
DescriptionReason:
- Lost contact at address %d.%d
Check:
1. Check InterBus-S bus
   at the above address
2. Restart the system

71108: InterBus-S module failure
DescriptionReason:
- The unit %s at address %d.%d
  reported internal unit error
Check:
1. Check the unit at
   the above address
2. Restart the system

71109: InterBus-S module failure
DescriptionReason:
- Incompatible definition
  of unit %s
  at address %d.%d!
Check:
1. Check the I/O system parameters
2. Restart the system

71110: InterBus-S module failure
DescriptionReason:
- Illegal address "%s" on unit %s
Check:
1. Change the address in the system
   parameters I/O unit configuration.

71111: Wrong product code.
DescriptionReason:
- The product code of unit %s
  doesn’t compare to any known id code.
Check:
1. Change product code in configuration.
2. or check module.

71112: Unequal # of units.
DescriptionReason:
- Definied number of units is different
  to the one binded to the board.
Check:
1. Change unit definition in
   the configuration.
2. or check the modules.

71113: IBS user command failure
DescriptionReason:
- Warning due to: %d
Check:
1. Check the InterBus-S configuration
   according to above error code.

71114: Invalid IP address
DescriptionReason:
'%s' is not a valid IP address
Check:
1.Check the Communication configuration.

71115: Invalid subnet mask
DescriptionReason:
'%s' is not a valid subnet mask
Check:
1. Check the Communication configuration

71116: Disable & Trust Level = 0
DescriptionReason:
Unit '%s' has been configured disabled
with Trust Level set to 0.
This is NOT allowed.
Check:
1. Change the configuration.

71117: Battery supervision res.
DescriptionReason:
The battery capacity = %d

71118: Battery supervision state
DescriptionReason:
The battery supervision is in wrong
state .
The state is = %s

71119: IO-unit table full.
DescriptionReason:
- Maximum number of IO-units
  of type %s exceeded.
  Max = %d
Check:
1. Reduce the number of IO-units.

71120: RAP option not installed.
DescriptionReason:
- The RAP Communication option has to be
  installed when configuring RAP or
  using SCWrite.
Check:
1. Reboot and install the
   RAP Communication option.

71121: RAP start-up failed.
DescriptionReason:
- The initialization of RAP failed.
Check:
1. Check internal log for decription.

71122: Incorrect IP address.
DescriptionReason:
- The address '%s'
  in protocol '%s'
  is not a correct IP address.
Check:
1. Change the address.

71123: No matching trans. prot.
DescriptionReason:
The transmission protocol '%s'
given for application protocol '%s'
could not be found.
Check:
1. Change the transmission protocol.

71124: Wrong trans prot. for NFS
DescriptionReason:
The transmission protocol for
the NFS protocol '%s' must be
TCP/IP.
Check:
1. Change the transmission protocol.

71125: Mount Permission denied
DescriptionReason:
Permission was denied to mount
the directory '%s'
on the server '%s'.
Check:
1. Change the User or Group ID.

71126: Directory not exported
DescriptionReason:
Mounting directory ’%s’
as ’%s’ failed
since it is not exported on the server
computer ’%s’,Protocol: ’%s’.
Check:
1. Export the directory on
the server computer.

71127: ID’s not the equal
DescriptionReason:
The User and Group ID’s has to have
the same value for all remote disks
Check:

71128: Ethernet option not inst.
DescriptionReason:
- The Ethernet Services option has to be
installed when remomte mounted disks
Check:
1. Reboot and install the Ethernet
Services option.

71129: Too many remote disks
DescriptionReason:
The maximum number of remote
mounted
disks is ’%d’.
Check:
1. Reduces the number of remote
mounted
disk.

71130: Too many remote servers
DescriptionReason:
The maximum number of servers for
remote mounted disks is ’%d’.
Check:
1. Reduces the number of servers

71131: Could not mount directory
DescriptionReason:
Mounting directory ’%s’
on the computer ’%s’ failed
Protocol: ’%s’.
Check:
1. Check the NFS server setup

71132: Battery supervision block
DescriptionReason:
The battery superviosion is blocked
becaurse a power off/on is made.
Will be released %s
Check:

71133: Battery not charged
DescriptionReason:
The battery is not charged after
the first charging state.
Check:
1. Check that the batlow signal is
connected to the DSQC347.
2. Check the battery connections
3. Check the battery.

71134: Parameter value not found
DescriptionReason:
Unit %s refers to values(%s)
that don’t exists.
Check:
1. Check unit configuration.

71135: Parameter defs. not found
DescriptionReason:
Unit type %s refers to
parameter definitions (%s)
that don’t exists.
Check:
1. Check unit type configuration.

71136: Parameter Error
DescriptionReason:
Parameter %s could not be
downloaded to unit %s
Check:
1. Check unit configuration.
2. Check precceding error messages

71137: Error code from unit
DescriptionReason:
The following error was returned:
%s
Check:
1. Check unit parameter configuration.

71138: Unknown Parameter
DescriptionReason:
%s is not a valid parameter
for unit %s
Check:
1. Check unit parameter configuration.

71139: Access error from IO
DescriptionReason:
- Cannot Read or Write signal %s
  due to communication down.
Check:
1. Check 'No contact with I/O unit'
   report for reason.

71140: Parameter act high error
DescriptionReason:
- Signal %s: This type of signal
  can't be set to active high.
Check:
1. Only output signals can be
   set to active high.

71141: Default out of range
DescriptionReason:
- The default value for
signal %s is out of range
Check:
1.Change the default value in
  configuration.

71142: Parameter Default error
DescriptionReason:
- Signal %s: This type of signal
  can't be assigned a default value
Check:
1. Only output signals can be assigned
default values.

71143: CTS/RTS not allowed
DescriptionReason:
- Serie channel %s:
  can't have option RTS/CTS.
Check:
1. Remove the option from configuration.

71144: Enable group failed
DescriptionReason:
- Failed to enable unit %s
  at address %d.%d
Check:
1. Check I/O system parameters
2. Check unit

71145: IBS bus deactivated
DescriptionReason:
- IBS bus changed into a none
  running mode.
Check:
1. Check the bus and restart the system

71146: Subscribe error
DescriptionReason:
- The maxlimit is less then minlimit
  maxlimit = %f minlimit = %f.
Check:
1. Make the correction and try again

71147: No response dsqc344
DescriptionReason:
- Access to the dsqc344 is denied
Check:
1. Check dsqc344 board
2. Check dsqc344 internal confiuration
3. Reduce cycle time on the slave unit

71148: No access to dsqc344
DescriptionReason:
- Access to the dsqc344 is denied
  due to %d
Check:
1. The dsqc344 is faulty. Replace board
2. Check dsqc344 internal confiuration

71149: IBS command failure
DescriptionReason:
- Incompatible definition
  due to %d!
Check:
1. Check the I/O system parameters
2. Restart the system

71150: The dsqc344 is busy
DescriptionReason:
- The response service capability
  is limited due to IBS configuration
Check:
1. Check dsqc344 internal confiuration
2. Change cycle time on the slave unit

71152: InterBus-S bus failure
DescriptionReason:
- %s bus failure.
  Lost contact at
  address %d.%d
Check:
1. Check InterBus-S bus
   at the above address
2. Restart the system

71153: Can't access parameter
DescriptionReason:
- Unit %s has parameters
   that can't be edited from
  the teach pendant.
Check:
1. Save the EIO configuration to
   a file
2. Use a text editor to change
   the parameters

71154: The unit has no power
DescriptionReason:
- Unit %s has no power connected
Check:
1. Attach power to the unit

71155: The unit is faulty
DescriptionReason:
- Unit %s has not been correctly
  connected
Check:
1. Diagnose with CMD application
2. Modify the dsqc344 loacal start-up
   procedure
3. Replace the dsqc344 firmware

71156: IPC queue full
DescriptionReason:
- The ipc queue %s was
  full, when sending to trap routine.
Check:
1. Restart the system

71157: Invalid app type
DescriptionReason:
- Com Unit %s: has an unspecified
  Com app type named: %s.
Check:
1.Check the Com app type against
  the one specified for the Com app

71158: Address out of range
DescriptionReason:
- The address of unit %s is out of
  range.
Check:
1.Change the address
2.Check the address syntax

71159: Signal Access illegal
DescriptionReason:
- Signal %s:
  The number of characters in
  Access %s
  is greater than %d or name missing.
Check:
1. Give a new Access that fits within
   the limits.

71160: Access level illegal
DescriptionReason:
- Signal %s:
  Access level %s in
  group %s is illegal.
- Legal choices all/man/auto/none.
Check:
1. Give a new access level for group
   that fits.

71161: Access name not found
DescriptionReason:
- Signal %s
  The Access name %s
  not found in EIO_USER_ACCESS.
Check:
1. Define %s in EIO_USER_ACCESS
   or change Access name.

71163: Signal on internal unit
DescriptionReason:
- Signal %s is not allowed
  to connect to %s
  because the unit is internal
Check:
1. Connect the signal to another unit

71164: Internal signal in cross
DescriptionReason:
- Signal %s is not allowed
  to be cross connected to
  signal %s
Check:
1. Make sure that none of the signals
   are internal.

71171: Illegal I/O Unit
DescriptionReason:
- The unit type %s used by unit
  %s is a non ABB device
Check:
1. Make sure that the I/O Plus Option
   is installed

8 Arcweld error messages

110001: Gas supervision
Check:
Check the welding equipment.

110002: Water supervision
Check:
Check the welding equipment.

110003: Arc supervision
Check:
Check the welding equipment.

110004: Voltage supervision
Check:
Check the welding equipment.

110005: Current supervision
Check:
Check the welding equipment.

110006: Wirefeed supervision
Check:
Check the welding equipment.

110007: Wirestick supervision
Check:
Check the welding equipment.

110008: Arc ignition failed
Check:
Check the welding equipment.

110009: Schedule transfer error
Check:
Define a weld schedule strobe input

110010: Schedule transfer error
Check:
The schedule port was busy with previous transfer.

110011: Process stopped
Check:
Process was stopped by digital input.

110012: Arc fill ignition failed
Check:
Check the welding equipment.

110013: Torch supervision
Check:
Check the welding equipment.

110021: Gas supervision
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110022: Water supervision
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110023: Arc supervision
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110024: Voltage supervision
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110025: Current supervision
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110026: Wirefeed supervision
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110027: Process stopped
Check:
Seam name: %.16s
Time from weld start: %.16s min
Process was stopped by digital input.

110028: Torch supervision
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110029: Arc ignition failed
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110030: Arc fill ignition failed
Check:
Seam name: %.16s
Time from weld start: %.16s min
Check the welding equipment.

110500: Arcitec welddata error
Loading or storing welddata failed.
Max allowed time for operation has expired.
Check:
Check external equipment.

110501: Arcitec welddata error
Storing welddata failed.
Check:
Check file path and external equipment.

110502: Arcitec welddata error
Loading welddata failed.
Check:
Check file path and external equipment.

110503: Switch is missing
One of the switch parameters %s or %s has to be defined.
Check:
Define one of the switch parameters.

110504: Value error
Argument %.16s must have an integer value.
Check:
The value of the argument must be an exact integer value. The current value has a fraction part. Change value.

110505: Argument error
Argument %s has a not allowed negative value.
Check:
Set argument %s to positive.

111000: Weave pattern error
Weave interpolation type error [Geometric = 0, Rapid = 1]
Check:
Adjust weave parameters

111001: Weave pattern error
Weave pattern shape error
[No shape = 0, Zig-zag shape = 1]
[V-shape = 2, Triangular shape = 3]
Check:
Adjust weave parameters

111002: Weave pattern error
Weave pattern cycle length error
(0 - 1) [m]
Check:
Adjust weave parameters

111003: Weave pattern error
Weave pattern cycle time error
(0 - 100) [s]
Check:
Adjust weave parameters

111004: Weave pattern error
Weave pattern width error
(0 - 1) [m]
Check:
Adjust weave parameters

111005: Weave pattern error
Weave pattern height error
(0 - 1) [m]
Check:
Adjust weave parameters

111006: Weave pattern error
Weave pattern left dwell error
(0 - 1) [m]
Check:
Adjust weave parameters

111007: Weave pattern error
Weave pattern center dwell error
(0 - 1) [m]
Check:
Adjust weave parameters

111008: Weave pattern error
Weave pattern right dwell error
(0 - 1) [m]
Check:
Adjust weave parameters

111009: Weave pattern error
Weave pattern bias error
(-1 - 1) [m]
Check:
Adjust weave parameters

111010: Weave pattern error
Weave pattern direction angle error
(-PI/2 - PI/2) [rad]
Check:
Adjust weave parameters

111011: Weave pattern error
Weave pattern tilt angle error
(-PI/2 - PI/2) [rad]
Check:
Adjust weave parameters

111012: Weave pattern error
Weave pattern rotation angle error
(-PI/2 - PI/2) [rad]
Check:
Adjust weave parameters

111013: Weave pattern error
Weave pattern horizontal offset error
(-1 - 1) [m]
Check:
Adjust weave parameters

111014: Weave pattern error
Weave pattern vertical offset error
(-1 - 1) [m]
Check:
Adjust weave parameters

111015: Weave pattern error
Weave pattern sync condition left error
(0 - 100) [%]
Check:
Adjust weave parameters

111016: Weave pattern error
Weave pattern sync condition right error
(0 - 100) [%]
Check:
Adjust weave parameters

111017: Weave pattern error
Forbidden combination of bias and shape
Bias only allowed for Zig-zag shape
Check:
Adjust weave parameters

111018: Weave pattern error
Forbidden combination of bias and width
Bias must be less than half the width
Check:
Adjust weave parameters

111019: Weave pattern error
Forbidden combination of dwells and cycle length
Dwells must be less than cycle length
Ramp slope (amplitude/length) is limited
Check:
Adjust weave parameters

112000: Board eipaw error
Incorrect digital output length
Check:
Change digital output length to 1

112001: Board eipaw error
Incorrect schedule port length
Check:
Change shedule port length to %.16s

112003: Board eipaw error
Incorrect wirefeeder port length
Check:
Change length

112004: Board eipaw error
Schedule number zero is not allowed.
Previous number will still be active.
Check:

113000: Equipment config error
Check:
AW and EIO configurations do not match

114000: Weldguide error
Check:
Check weldguide parameters and equipment

115000: Arcitec Data Error
Invalid parameter_id: %.16s
detected.
Check:
Check Arcitec configuration file or Power Source external communication.

115001: Arcitec Data Error
Invalid unit_id: %.16s
detected.
Check:
Check Arcitec configuration file or Power Source external communication.

115002: Arcitec Data Error
Invalid transmission length:
%.16s detected.
Check:
Check Arcitec configuration file or Power Source external communication.

115003: Arcitec Data Error
Invalid selection_id: %.16s
detected.
Check:
Check Arcitec configuration file or Power Source external communication.

115004: Arcitec Data Error
Arcitec systems with different units.
%.16s and %.16s.
Check:
Check Arcitec configuration file.

115005: Arcitec Data Error
Units not defined for Arcitec system.
Check:
Check Arcitec configuration file.

115006: Arcitec Data Error
Illegal number: %.16s
of tuning parameters.
Check:
Check Arcitec configuration file.

116000: Track error
Check:
Check joint definition

116001: Track start error
Check:
Check joint definition

116002: Track max path corr error
Check:
Check joint definition

116003: Track communication error
Check:
Check hardware

117001: Welding equipment error
EPROM checksum error in Welddata Unit detected at power up.
Check:
EPROM in Welddata Unit is faulty. Running with this error gives unpredictable result.
Exchange EPROM.

117002: Welding equipment error
Internal RAM read/write error in Welddata Unit detected at power up.
Check:
At least one memory cell in internal microprocessor memory failed in read /write test. Running with this error gives unpredictable result.
Replace Welddata Unit.

117003: Welding equipment error
External RAM read/write error in Welddata Unit detected at power up.
Check:
At least one memory cell in external microprocessor memory failed in read /write test. Running with this error gives unpredictable result.
Replace Welddata Unit.

117004: Welding equipment error
DC supply voltage for 5 Volt regulator in Welddata Unit has been down.
Check:
Indicates that there is a problem in power supply but the function is probably not affected. Check incoming power supply to Welddata Unit.

117012: Welding equipment error
Welddata Unit CAN-controller for internal bus is in WARNING state.
Check:
Change data several times or reset welding equipment with power switch. If the error do not disappear, check bus connections and/or exchange Welddata Unit.

117013: Welding equipment error
Welddata Unit CAN-controller for external bus is in WARNING state.
Check:
Change data several times or reset welding equipment with power switch. If the error do not disappear, check bus connections and/or exchange Welddata Unit.

117014: Welding equipment error
Welddata Unit CAN-controller for internal bus is in BUS-OFF state.
Check:
Reset welding equipment with power switch. If the error do not disappear, check bus connections and/or exchange Welddata Unit.

117015: Welding equipment error
Welddata Unit has detected that a received internal CAN message was lost (Overwritten by a later message).
Check:
Reset welding equipment with power switch.

117016: Welding equipment error
Welddata Unit has detected that a received external CAN message was lost (Overwritten by a later message).
Check:
Reset welding equipment with power switch.

117017: Welding equipment error
Welddata Unit lost contact with Wirefeed Unit.
Check:
Check connection cable between Welddata
Unit and wirefeed control board, check power supply to wirefeed control board.

117018: Welding equipment error
Welddata Unit has lost contact with Olivia Unit.
Check:
Check connection cable between Welddata Unit and Olivia unit, check power supply to Olivia Unit

117019: Welding equipment error
Non-volatile RAM data value failure detected in Welddata Unit at power up. Checksum error.
Check:
Probably caused by low memory backup battery voltage. Welding equipment will be reset to a default state. Data in Welddata Unit will be lost.
Possible to run without limitations.

117020: Welding equipment error
Non-volatile RAM data value failure detected in Welddata Unit at power up. Non numeric setting parameter out of range.
Check:
Welding equipment will be reset to a default state. Data in Welddata Unit will be lost. Possible to run without limitations.

117021: Welding equipment error
Invalid combination of non-numeric setting parameters in Welddata Unit detected at power up.
Check:
Welding equipment will be reset to a default state. Data in Welddata Unit will be lost. Reset welding equipment with power switch.

117022: Welding equipment error
CAN-bus (external) transmit buffer overflow in Welddata Unit.
Check:
Welddata Unit are unable to transmit data at the needed rate. Could be caused by unnormal occupation on the bus. Reset welding equipment with power switch.

117023: Welding equipment error
CAN-bus (external) receive buffer overflow in Welddata Unit.
Check:
Welddata Unit are unable to process received messages at the needed rate. Reset welding equipment with power switch.

117024: Welding equipment error
Fragments not in number order when Welddata Unit received a fragmented message.
Check:
The parts of a fragmented message were not received in proper order. A weld data block transmission has been faulty received. Reset welding equipment with power switch.

117025: Welding equipment error
Incompatible format of weld data block. Welddata Unit received data that is stored in another program version with other format version.
Check:
Find data with correct version or enter new data.

117026: Welding equipment error
Program execution error.
Watch dog in Welddata Unit program activated.
Check:
Reset welding equipment with power switch.

117027: Welding equipment error
Undocumented Welddata Unit error.
Check:
Request additional information from ESAB/ABB.

117028: Welding equipment error
Undocumented Welddata Unit error.
Check:
Request additional information from ESAB/ABB.

117029: Welding equipment error
Undocumented Welddata Unit error.
Check:
Request additional information from ESAB/ABB.

117201: Welding equipment error
EPROM checksum error in Powersource Control Unit.
Check:
EPROM in Powersource Control Unit is faulty. Running with this error gives unpredictable result.
Replace EPROM.

117202: Welding equipment error.
Internal RAM read/write error in Powersource Control Unit detected at power up.
Check:
At least one memory cell in internal microprocessor memory failed in read /write test. Running with this error gives unpredictable result.
Replace Powersource Control Unit.

117204: Welding equipment error
DC supply voltage to 5 Volt regulator in Powersource Control Unit has been down.
Check:
Indicates that there is a problem in power supply but the function is probably not affected. Check incoming power supply to Powersource Control Unit

117205: Welding equipment error
High DC inverter bus voltage. Hardware will shut down inverter till voltage comes down to normal.
Check:
Might be caused by high mains impedance or transients, possible to restart welding as soon as voltage has dropped below limit.

117206: Welding equipment error
Temperature in power source heatsink too high. Inverter is shut down until temperature switch is closed again.
Check:
Ensure that there is no obstacle that reduces the cooling airflow that passes through the heatsink of the powersource. Wait until temperature switch is closed.

117207: Welding equipment error
High current in inverter circuit. Might be caused by component failure.
Check:
Reset welding equipment with power switch. Check that the power source does not consume unnormal high current without start command. If so: there is a component failure.

117208: Welding equipment error
PCB supply voltage 15VC on Powersource Control Unit to high or to low.
Check:
Replace Powersource Control Unit.

117209: Welding equipment error
PCB supply voltage -15V on Powersource Control Unit to high or to low.
Check:
Replace Powersource Control Unit.

117210: Welding equipment error
PCB supply voltage 15VB on Powersource Control Unit to high or to low.
Check:
Replace Powersource Control Unit.

117211: Welding equipment error
Long term difference between requested and actual weld current value.
Check:
Hardware problem in current servo system
(Power source control board or inverter block) or unnormal load conditions
(= bad welding

117212: Welding equipment error.
Internal CAN communication failure CAN
circuits in Powersource Control Unit is in WARNING state.
Check:
Change data several times or reset welding equipment with power switch. If the error do not disappear, check bus connections and/or exchange Power-source
Control Unit.

117215: Welding equipment error
Powersource Control Unit has detected that a received internal CAN message was
lost (overwritten by a later message).
Check:
Reset welding equipment with power switch.

117226: Welding equipment error
Program execution error.
Watch dog in Powersource Control Unit program activated.
Check:
Reset welding equipment with power switch.

117227: Welding equipment error
Undocumented Powersource Control Unit
error.
Check:
Request additional information from ESAB/ABB.

117228: Welding equipment error
Undocumented Powersource Control Unit error.
Check:
Request additional information from ESAB/ABB.

117229: Welding equipment error
Undocumented Powersource Control Unit error.
Check:
Request additional information from ESAB/ABB.

117301: Welding equipment error
EPROM checksum error in Wirefeed unit detected at power up.
Check:
EPROM in Wirefeed unit is faulty.
Running with this error gives unpredictable result.
Exchange EPROM.

117302: Welding equipment error
Internal RAM read/write error in Wirefeed Unit detected at power up.
Check:
At least one memory cell in internal microprocessor memory failed in read /write test. Running with this error gives unpredictable result.
Replace Wirefeed Unit.

117304: Welding equipment error
DC supply voltage for 5 Volt regulator in Wirefeed Unit has been down.
Check:
Indicates that there is a problem in power supply but the function is probably not affected. Check incoming power supply to Wirefeed Unit.

117308: Welding equipment error
PCB supply voltage 15V on Wirefeed Unit to high or to low.
Check:
Replace Wirefeed Unit.

117309: Welding equipment error
PCB supply voltage 24V on Wirefeed Unit
to high or to low.
Check:
Replace Wirefeed Unit.

117311: Welding equipment error
Long term difference between requested and actual wirefeed velocity.
Check:
Hardware problem in wirefeed servo system or voltage drop in 42 V AC supply.

117312: Welding equipment error
Internal CAN communication failure CAN circuits in Wirefeed Unit is in WARNING state.
Check:
Change wirefeed speed several times or reset welding equipment with power switch. If the error do not disappear, check bus connections and/or exchange Wirefeed Unit.

117315: Welding equipment error
Wirefeed Unit has detected that a received internal CAN message was lost (overwritten by a later message).
Check:
Reset welding equipment with power switch.

117326: Welding equipment error
Program execution error.
Watch dog in Wirefeed Unit program activated.
Check:
Reset welding equipment with power switch.

117327: Welding equipment error
Undocumented Wirefeed Unit error.
Check:
Request additional information from ESAB/ABB.

117328: Welding equipment error
Undocumented Wirefeed Unit error.
Check:
Request additional information from ESAB/ABB.

117329: Welding equipment error
Undocumented Wirefeed Unit error.
Check:
Request additional information from ESAB/ABB.

117500: File error
Failed open file.
File name unknown.
Check:
Check file name.

117501: File error
Failed writing to file.
Check:
Check file name.

117502: File error
Failed reading from file.
Check:
Check file name.

117503: Illegal schedule number
Schedule number %.16s
is not allowed.
Check:
Change schedule number.

117504: Communication error
Message header form Arcitec power source unknown.
Check:
Check connection to power source.
Check Arcitec configuration file.

117505: Communication error
Arcitec power source is not responding.
Check:
Check connection to power source.

117506: Communication error
Messages from Arcitec power source are not possible to evaluate.
Messages are too short.
Check:
Check connection to power source.
Check Arcitec configuration file.

118000: Ext CAN com failure
Too many requests without response
Check:
Check communication configuration

118500: Arcitec tuning error
There are no Arcitec system defined.
Check:
Define Arcitec system or do not write towards Power Source.

118501: Arcitec tuning error
Parameter does not exist.
Check:
Change parameter identity.

118502: Arcitec tuning error
Parameter must be numerical.
Check:
Change parameter identity.

118503: Arcitec tuning error
Parameter value outside limits.
Check:
Change parameter value.

119000: Installation error
External axes are not allowed in this type of robot configuration
Check:
Remove mechanical units

9 Spotweld error messages

120001: Spot weld system error
Spot weld proc not idle
Check:
Set the process state defined by SwInit to idle

120002: Spot weld system error
Parameter %s

120003: SwStart Timeout negative

120004: SwInit Interrupt negative

120005: ProcId. The reason is either:
-ProcId does not correspond to the value given from SwInit
-The spot weld process has been cancelled

120006: Spot weld comm. error
Reason: %s

120007: Response slower than poll rate

120008: No more BOSCH connection available

120010: Spot gun error
Reason:%s

120011: Spot gun bad config.
Reason: %s

120012: IO signal missing
Check:
1.Configure the mandatory signals used

120013: PERS var missing
the swtimer.sys module doesn’t fit with this kind of gun
Check:
2.Replace it

120014: Error Number %d
Check:
See Nc gun manual

120057: Gun jog. forbidden
Gun jogging is not allowed when axis not synchronised
Check:
1.Axis calibration state

120058: Gun Calibration Ok

120060: Gun Position Error
Gun can’t reach the position reference
Check:
1.The drive for power on
2.The encoder
3.Mechanical parts or stuck tips

120061: Gun refer. overrun
Swit timer can’t consume the position reference received
Check:
1.Swit timer is welding and a move instr. is executing concurrently

120062: Gun Force error
Gun can’t apply the force reference
Check:
1.Make a gun init
2.Drive and force sensor calibration

120063: Gun Refer. underrun
Swit timer has no more reference to consumme
Check:
1.Serial link and comm log error

120064: Gun Force sensor fault
force sensor out of order
Check:
1.Force sensor wiring
2.Change sensor

120065: Gun motor overcurrent
motor current too great
Check:
1.Drive calibration
2.Mechanical unit or obstacle

120066: Gun reference fault
gap between references too great
Check:
1.Robot or gun calibration
2.Gun position

120067: Gun command not allowed
Motion forbidden during an action
Check:
1.Wait until the end of the action before ordering new one

120070: Gun drive fault
fault detected by drive unit
Check:
1.drive unit leds

120075: Gun encoder error
Check:
1.Encoder wring

120084: Gun wrong pos computed
inconsystency between revolution counter and encoder value
Check:
1.Drive and swit measurement bus
2.Make a gun first init

120085: Gun revol. counter error
Check:
1.Encoder wiring

120090: Gun not calibrated
No calibration done or lost
Check:
1.Make a gun first init

10 Paint error messages.

130001: Equipment error
Paint process and motion stopped.
Check:
Check the paint equipment.

130002: Equipment error
The argument is not a persistent variable.
Check:
Change the argument to a persistent.

130003: Trig plane error
In PaintL %s: One trig plane, %s, are defined outside the programmed path.
Check:
Change eventdata or reprogram path.

130004: Trig plane error
Trig plane error
In PaintL %s: Two trig planes, %s and %s, are defined outside the programmed path.
Check:
Change eventdata or reprogram path.

130005: Trig plane error
In PaintL %s: Three trig planes, %s, %s, and %s, are defined outside the programmed path.
Check:
Change eventdata or reprogram path.

130006: Trig plane error
In PaintL %s: Four trig planes, %s, %s, %s and %s, are defined outside the programmed path.
Check:
Change eventdata or reprogram path.

130007: Trig plane error
In PaintL %s: Four trig planes, %s, %s, %s and %s and more are defined outside the programmed path.
Check:
Change eventdata or reprogram path.

131000: Argument error.
The argument is not an integer.
Check:
Change the argument to an integer.

131001: Argument error.
The argument is not an array.
Check:
Change the argument to an array.

131002: Argument error.
The argument is not a persistent variable.
Check:
Change the argument to a persistent.

131003: Argument error.
The array argument has too many dimensions.
Check:
Change the array to one dimension.

131004: Brush table error.
Only brush table %s is allowed.
Check:
Change to allowed brush table.

131005: Brush number error.
Only brush numbers less than or equal to %s is allowed.
Check:
Change to allowed brush number.

132000: Brush number error.
The brush number is outside the limits for the activated brush table.
Check:
Change argument within limits.

132001: Brush table error.
There are no brush table activated.
Check:
Activate a brush table.

CONTENTS

Page

1 Simple Material Handling 3
1.1 What the robot does 3
1.2 The main routine 3
1.3 Operating the gripper 3
1.4 Fetching a part from the In feeder 4
1.5 Leaving the part in the machine 4
1.6 Starting to process 5
1.7 Fetching the part from the machine 5
1.8 Leaving the part on the Out feeder 5
2 Material Handling 7
2.1 What the robot does 7
2.2 The main routine 7
2.3 Operating the gripper 8
2.4 Starting production 9
2.5 Fetching the part from the In feeder 9
2.6 Leaving the part in the machine 9
2.7 Updating operating statistics 10
2.8 Stopping production for the day 10

1 Simple Material Handling

1.1 What the robot does

The robot takes parts to and from a machine, as in Figure 1.

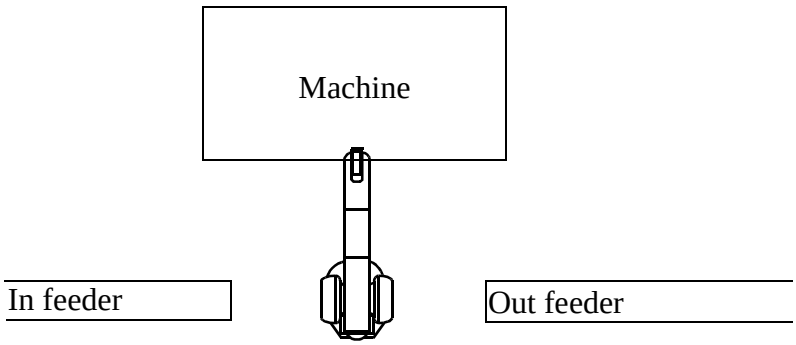


Figure 1 The robot gives a part to a machine which then processes it.

First, the robot fetches a part from the In feeder and places it in the machine where the part is processed. Then, when this has been done, the robot takes the part and places it on the Out feeder.

This work cycle is repeated until the operator stops production.

1.2 The main routine

The main routine is built up of a number of routine calls which reflect the robot work cycle.

Routine *main*	Comments
fetch_part;	Fetch part from In feeder.
leave_machine;	Leave the part in the machine.
process_part;	Start the actual processing.
fetch_machine;	Fetch the part.
leave_part;	Leave the part on the Out feeder.

1.3 Operating the gripper

The robot is equipped with a gripper that handles parts. A tool, *tool1*, and its associated tool centre point (TCP), is defined for this.

The tool is controlled by a digital output signal defined in the system parameters with the name *gripper*. A high signal indicates that the gripper is holding the part, and a low signal indicates that the part has been released.

In addition, a load data, *load1*, is defined which describes the load held by the gripper. The best possible motion performance is achieved if the correct load is always specified.

As the gripper grips and releases parts several times during the course of the program, it is best to set up separate routines for this which can be called by the program.

Routine *grip*	Comments
Set gripper;	Grip the part.
WaitTime 0.3;	Wait 0.3 s.
GripLoad load1;	Specify that there is a load in the gripper.

Routine *release*	Comments
Reset gripper;	Release the part.
WaitTime 0.3;	Wait 0.3 s.
GripLoad LOAD0;	Specify that there is no load in the gripper

1.4 Fetching a part from the In feeder

A part is fetched from the In feeder. As the robot cannot go straight from the previous position (Out feeder), it performs a joint movement to the first position. Then, it uses linear movement to achieve good path accuracy.

Routine *fetch_part*	Comments
MoveJ *, vmax, z50, tool1;	Go quickly to position near In feeder.
MoveL *, v1000, z30, tool1;	Go to position above part.
MoveL *, v200, fine, tool1;	Go slowly to grip position.
grip;	Grip part.
MoveL *, v200, z30, tool1;	Go to position above part.

1.5 Leaving the part in the machine

The robot leaves the part in the machine and then leaves that area so that the machine can be started.

Routine *leave_machine*	Comments
MoveJ *, vmax, z50, tool1;	Go quickly to position outside machine.
MoveL *, v500, z10, tool1;	Go to machine.
MoveL *, v200, fine, tool1;	Go to leave position.
release;	Release part.
MoveL *, v200, z30, tool1;	Go to position above part.
MoveL *, v500, z30, tool1;	Go to position above machine.

1.6 Starting to process

Processing starts when the robot pulses an output, *do1*. Then, using the input *di1*, the machine informs the robot that the part has been processed and can be fetched.

Routine *process_part*	Comments
PulseDO do1;	Pulse output to start machine.
WaitDI di1, 1;	Wait for the ready signal.

1.7 Fetching the part from the machine

The robot fetches the part from the machine.

Routine *fetch_machine*	Comments
MoveL *, v500, z10, tool1;	Go to machine.
MoveL *, v200, fine, tool1;	Go to fetch position.
grip;	Grip part.
MoveL *, v200, z30, tool1;	Go to position above part.
MoveL *, v500, z30, tool1;	Go to position outside machine.

1.8 Leaving the part on the Out feeder

The robot leaves the part on the Out feeder.

Routine *leave_part*	Comments
MoveJ *, vmax, z30, tool1;	Go quickly to position near Out feeder.
MoveL *, v500, z30, tool1;	Go to position above part.
MoveL *, v200, fine, tool1;	Go slowly to leave position.
release;	Release part.
MoveL *, v200, z30, tool1;	Go to position above part.

2 Material Handling

2.1 What the robot does

The robot takes parts to and from a machine, as in Figure 2.

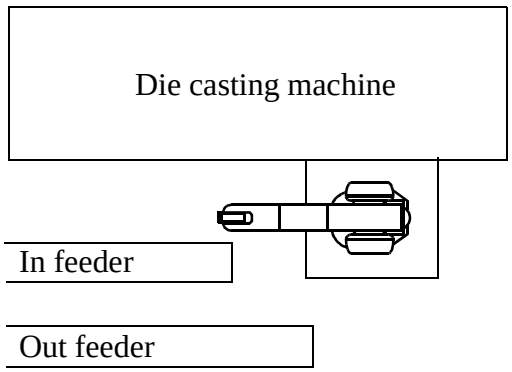


Figure 2 The robot serves a machine.

First, the robot fetches a part from the In feeder and places it in the machine. When the machine is ready, the robot grips the part and places it on the Out feeder.

The work cycle is repeated until the operator presses a push-button “Stop production”. The robot then completes the cycle, but does not fetch any new part from the In feeder.

The robot keeps a record of production statistics; it displays the number of parts produced during the day on the teach pendant and also, at the end of the work day, stores this information on a diskette that can be read using a PC.

2.2 The main routine

The main routine is built up of a number of routine calls which reflect the robot work cycle.

A digital input signal, *prodstop*, defined in the system parameters, is used to find out if the button “Stop production” is depressed. The button remains depressed until someone presses it again.

Routine *main*	Comments
start_production;	Initialise production for the day.
WHILE Dinput(prodstop) = 0 DO	Repeat the cycle until the button is pressed.
fetch_part;	Fetch the part from In feeder.
leave_machine;	Leave part in the machine.
process_part;	Start the actual processing.
fetch_machine;	Fetch the part.
leave_part;	Leave the part on Out feeder.
update_cycle;	Update operating statistics.
ENDWHILE	
stop_production;	Stop production for the day.

The routines *process_part*, *fetch_machine* and *leave_part* are not included in this example.

2.3 Operating the gripper

A tool, *gripper1*, defines the TCP and the weight of the gripper. This tool data is defined in the system module USER. In this way, the tool is always present in memory irrespective of which program is loaded.

The gripper is controlled by electric, bistable air valves, which means that there is one signal that controls the grip action and another that controls the release. The names of the signals are defined in the system parameters as *grip1* and *release1*. There is also a signal, *gripok*, that is high if a part is held by the gripper. This signal is used to check if the gripper has gripped a part correctly.

A load data, *payload*, is defined which describes the load held by the gripper. The best possible motion performance is achieved if the correct load is always specified.

As the gripper grips and releases parts several times during the course of the program, it is best to set up separate routines for this which can be called by the program. For example:

Routine *grip_part*	Comments
Reset release1;	
Set grip1;	Grip the part.
WaitTime 0.5;	Wait 0.5 s.
IF DInput(gripok)=0 THEN	If error (no part in the gripper) ...
TPWrite "ERROR: No part in the gripper";	Write error message on teach pendant.
EXIT;	Exit program execution.
ENDIF	
GripLoad payload;	Specify that there is a load.

The routine *release_part* is not included in this example.

2.4 Starting production

Before the actual production is started, the counter (*reg1*), which counts the number of parts that are produced during the day, is set to zero. The robot also goes into a home position.

Routine *start_production*	Comments
reg1 := 0;	Reset the counter.
MoveJ home, v500, fine, gripper1;	Go to home position.

In this example, all positions (e.g. *home* or *p1*) are named. They are stored as separate position data and can thus be reused in subsequent instructions. However, it is often just as easy to store the positions directly in the instructions (indicated by * in the instruction).

2.5 Fetching the part from the In feeder

Before fetching a part, the robot must check if there is any part to fetch. It does this by means of a photocell (via the *feeder* signal). This informs the robot if there is a part in position or not. If there is no part, the operator is sent a message and must first correct the error before starting program execution again.

Routine *fetch_part*	Comments
WHILE DInput(feeder) = 0 DO	Check if there is any part to fetch.
TPErase;	If not: Clear the teach pendant and
TPWrite "ERROR: No part on feeder";	write error message. Then wait until
TPWrite "";	the start signal is given by the operator.
TPReadFK reg2, "Put part on feeder and press start", "Start", "", "","","",";	
ENDWHILE	
MoveJ p1, vmax, z50, gripper1;	Go quickly to position above part.
MoveL p2, v100, fine, gripper1;	Go to grip position.
grip_part;	Grip part.
MoveL p1, v200, z30, gripper1;	Go to position above part.

2.6 Leaving the part in the machine

The robot leaves the part in the machine and then leaves that area so that the machine can be started. Often, the robot and the machine communicate with one another to check such things as whether the machine is open. This check is not included in the following example.

Routine *leave_machine*	Comments
MoveJ p3, vmax, z50, gripper1;	Go quickly to position outside machine.
MoveL p4, v500, z10, gripper1;	Go in to machine.
MoveL p5, v100, fine, gripper1;	Go to release position.
release_part;	Release part.
MoveL p4, v200, z30, gripper1;	Go to position above part.
MoveL p3, v500, z50, gripper1;	Go to position outside machine.

2.7 Updating operating statistics

The number of parts produced during the day is written on the teach pendant display.

Routine *update_cycle*	Comments
reg1 := reg1 +1;	Increment produced parts.
TPErase;	Clear the display.
TPWrite "";	A few blank lines.
TPWrite "";	
TPWrite "No of produced parts = " \Num:=reg1;	The number of parts.

2.8 Stopping production for the day

If the operator presses “Stop production” and the robot has completed a work cycle, the robot goes to home position. In addition, the production figures for the day (the day’s date followed by the number of parts produced) are written on diskette.

Routine *stop_production*	Comments
MoveJ home, v500, fine, gripper1;	Go to home position.
Open "flp1:" \File:="logfile.doc", file\Append;	Open the file for writing.
Write file, CDate() \Num:=reg1;	Write to the file.
Close file;	Close the file.
Stop;	Stop program execution.

Before a file can be opened, the data, *file*, must be created by the type *iodev*. The real name of the file is *logfile.doc*.

CONTENTS

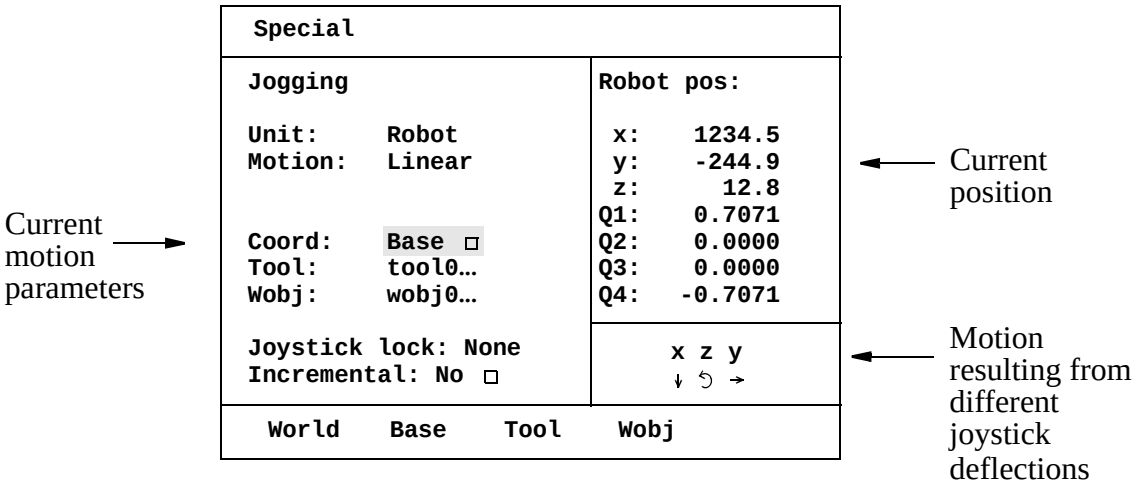
Page

1 The Jogging Window 3
1.1 Window: Jogging 3
1.1.1 Menu: Special 3
2 The Inputs/Outputs Window 4
2.1 Window: Inputs/Outputs 4
2.1.1 Menu: File 4
2.1.2 Menu: Edit 5
2.1.3 Menu: View 5
3 The Program Window 6
3.1 Moving between different parts of the program 6
3.2 General menus 7
3.2.1 Menu: File 7
3.2.2 Menu: Edit 8
3.2.3 Menu: View 9
3.3 Window: Program Instr 10
3.3.1 Menu: IPL1 (shows different instruction pick lists) 10
3.3.2 Menu: IPL2 (shows different instruction pick lists) 10
3.4 Window: Program Routines 11
3.4.1 Menu: Routine 12
3.4.2 Menu: Special 12
3.5 Window: Program Data 13
3.5.1 Menu: Data 13
3.5.2 Menu: Special 14
3.6 Window: Program Data Types 15
3.6.1 Menu: Types 15
3.7 Window: Program Test 16
3.7.1 Menu: Special 17
3.8 Window: Program Modules 18
3.8.1 Menu: Module 18
4 The Production Window 19
4.1 Window: Production 19
4.1.1 Menu: File 19
4.1.2 Menu: Edit 19
4.1.3 Menu: View 20
5 The FileManager 21
5.1 Window: FileManager 21

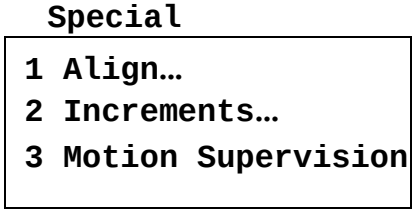
5.1.1 Menu: File 21
5.1.2 Menu: Edit 22
5.1.3 Menu: View 22
5.1.4 Menu: Options 22
6 The Service Window 23
6.1 General menus 23
6.1.1 Menu: File 23
6.1.2 Menu: Edit 24
6.1.3 Menu: View 25
6.2 Window Service Log 26
6.2.1 Menu: Special 26
6.3 Window Service Calibration 27
6.3.1 Menu: Calib 27
6.4 Window Service Commutation 28
6.4.1 Menu: Com 28
7 The System Parameters 29
7.1 Window: System Parameters 29
7.1.1 Menu: File 29
7.1.2 Menu: Edit 30
7.1.3 Menu: Topics 30
7.1.4 Menu: Types 31

1 The Jogging Window

1.1 Window: Jogging



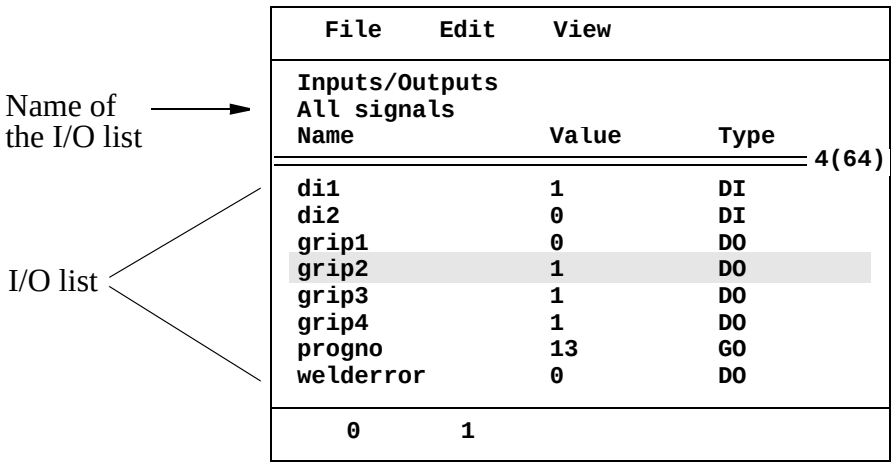
1.1.1 Menu: Special



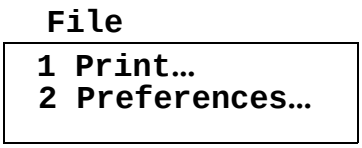
Command	Used to:
Align	Align the tool (see page 6-9)
Increments	Specify the sizes of the user defined increments (see page 6-14)
Motion Supervision	Turn Motion Supervision on/off (see page 6-5)

2 The Inputs/Outputs Window

2.1 Window: Inputs/Outputs

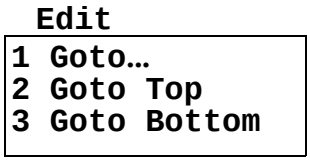


2.1.1 Menu: File



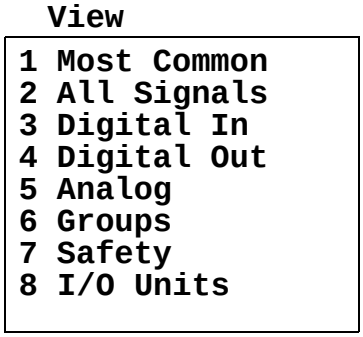
Command	Used to:
Print	print the current I/O list (see page 7-8)
Preferences	make preferences in the Inputs/Outputs window (see page 7-4)

2.1.2 Menu: Edit



Command:	Used to:
Goto	go to a specific line in the list
Goto Top	go to the first line in the list
Goto Bottom	go to the last line in the list

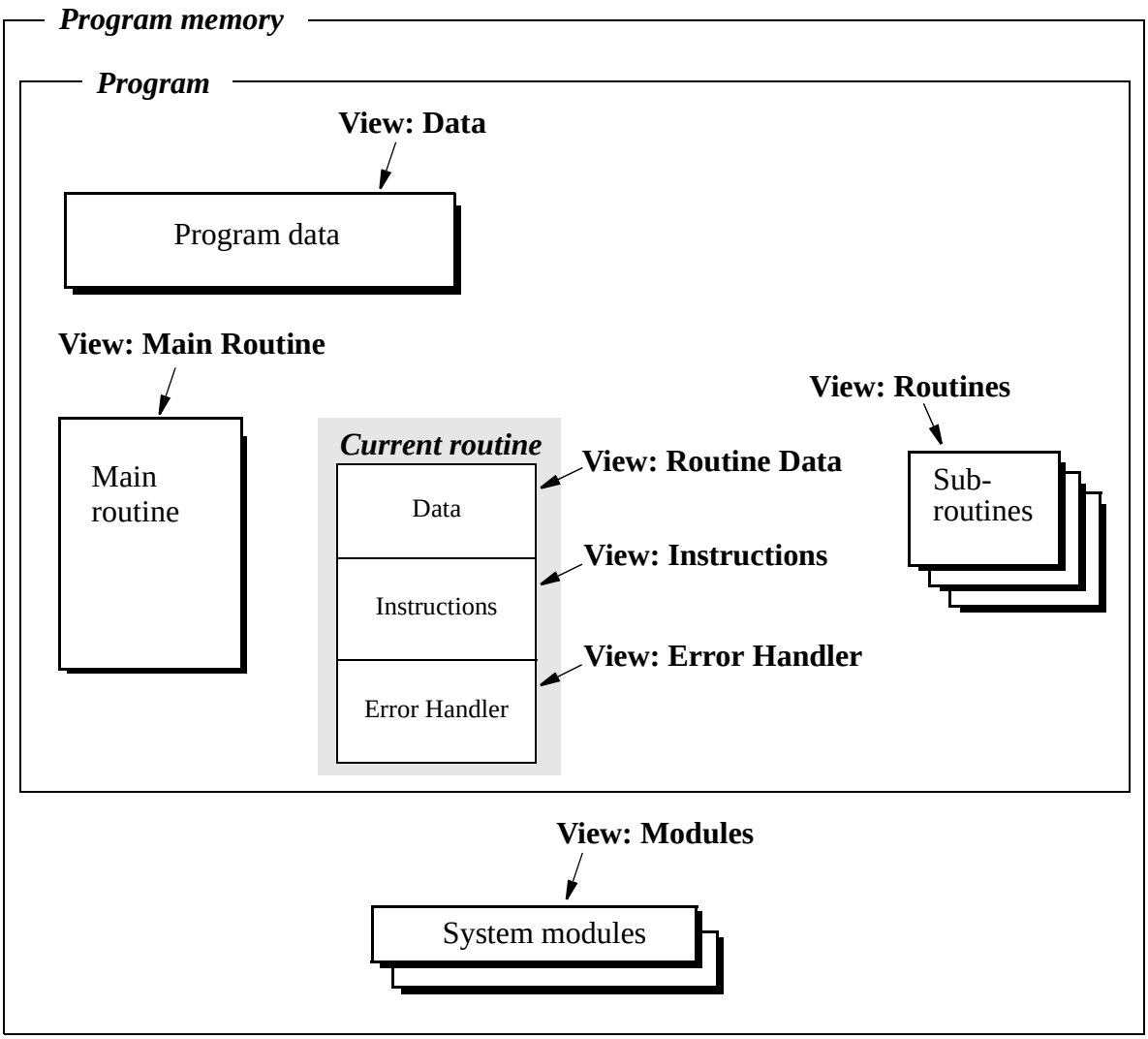
2.1.3 Menu: View



Command:	Used to view: (see page 7-4)
Most Common	the most common list
All Signals	all user signals
Digital In	all digital inputs
Digital Out	all digital outputs
Analog	all analog signals
Groups	all groups of digital signals
Safety	all safety signals
I/O Units	all I/O units

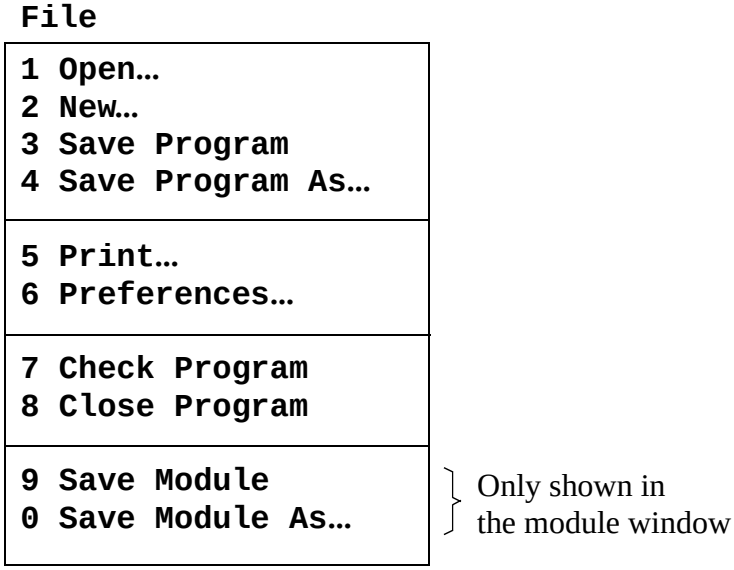
3 The Program Window

3.1 Moving between different parts of the program



3.2 General menus

3.2.1 Menu: File



Command:	Used to:
Open	read programs from mass storage (see page 8-7)
New	create new programs (see page 8-6)
Save Program	save programs on mass storage (see page 8-28)
Save Program As	save programs on mass storage with new names (see page 8-28)
Print	print the program (see page 8-29)
Preferences	make preferences in the Program window (see page 8-60)
Check Program	check that the program is correct (see page 8-21)
Close Program	erase the program from the program memory
Save Module	save a module on mass storage (see page 8-58)
Save Module As	save a module on mass storage with a new name (see page 8-58)

3.2.2 Menu: Edit

Edit

```
  Undo “Latest action”
1 Cut
2 Copy
3 Paste
4 Goto Top
5 Goto Bottom
6 Mark
7 Change Selected
8 Value
9 ModPos
0 Search…
  Show/Hide IPL
```

Command	Used to:
Undo	perform an undo on the latest action possible to undo in selected window (see page 8-36)
Cut	cut selected lines to the clipboard buffer (see page 8-21)
Copy	copy selected lines to the clipboard buffer (see page 8-21)
Paste	paste the contents of the clipboard buffer into a program (see page 8-21)
Goto Top	go to the first line (see page 8-30)
Goto Bottom	go to the last line (see page 8-30)
Mark	select several lines (see page 8-30)
Change Selected	change an instruction argument (see page 8-33)
Value	show the current value (for the selected argument) (see page 8-50)
ModPos	modify a position (see page 8-31)
Search	search for/replace a specific argument (see page 8-37)
Show/Hide IPL	show/hide an instruction pick list (see page 8-15)

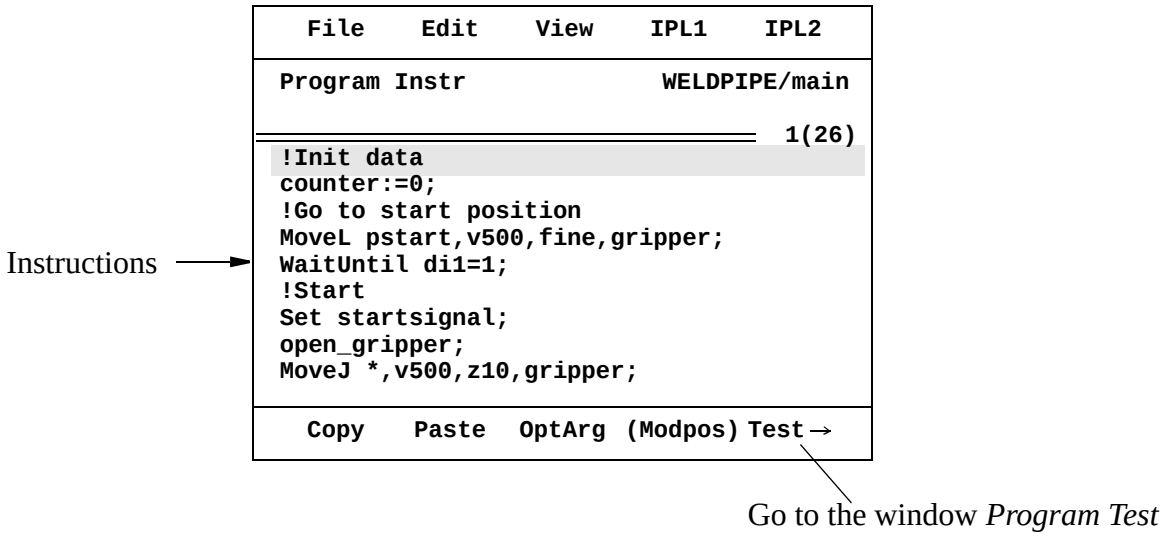
3.2.3 Menu: View

View

1 Instr.”<latest routine>” **2 Routines** **3 Data ”<latest type>”** **4 Data Types** **5 Test** **6 Modules**
7 Main Routine **8 Selected Routine**
9 Error Handler

Command	Used to view:
Instr.	instructions for the current routine – *Program Instruction* window – (see page 8-12)
Routines	all routines – *Program Routines* window – (see page 8-9)
Data	program data – *Program Data* window – (see page 8-45)
Data Types	all data types – *Program Data Types* window – (see page 8-45)
Test	the *Program Test* window (see page 8-22)
Modules	all modules – *Program Modules* window – (see page 8-55)
Main Routine	instructions for the main routine (see page 8-12)
Selected Routine	instructions for the selected routine (see page 8-12)
Error Handler	error handler of the current routine (see page 8-52)

3.3 Window: Program Instr



3.3.1 Menu: IPL1 (shows different instruction pick lists)

IPL1

```
1 Common
2 Prog. Flow
3 Various
4 Motion Settings
5 Motion&Process
6 IO
7 Communicate
8 Interrupts
9 Error Recovery
0 System&Time
  Mathematics
```

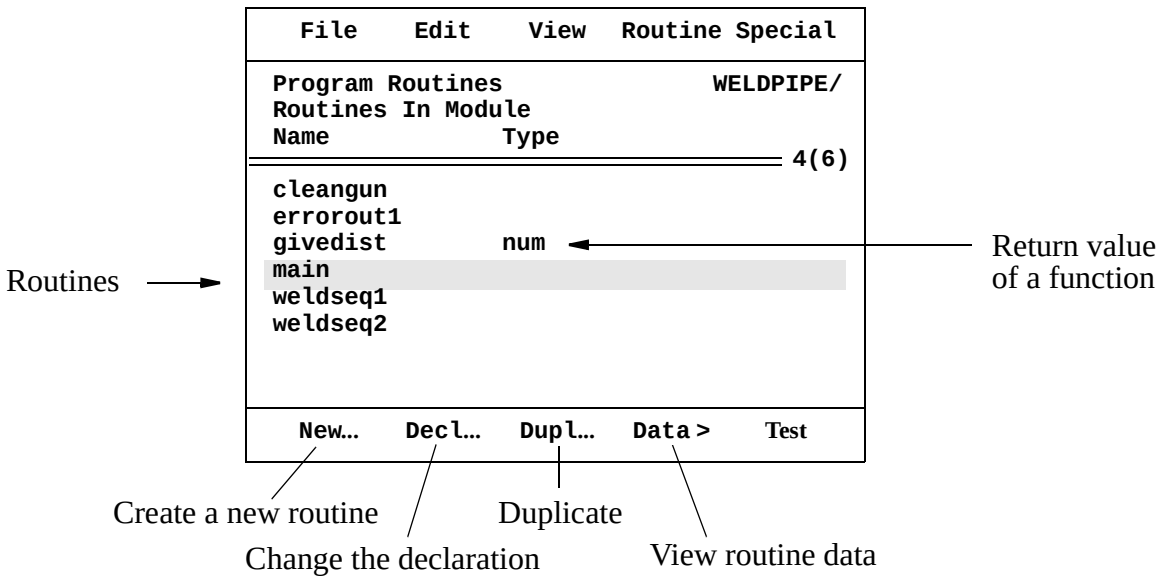
3.3.2 Menu: IPL2 (shows different instruction pick lists)

IPL2

```
1 Most Common 1
2 Most Common 2
3 Most Common 3
```

```
4 Motion Set Adv
5 Motion Adv
6 Ext. Computer
7 Multi Tasking
8 RAPID Support
9 Service
```

3.4 Window: Program Routines



3.4.1 Menu: Routine

Routine

1 Routine Data
2 Instructions **(Error Handler)** **(Backward Handler)**
5 In Module **6 In System**
7 Add Error Handler **8 Add Backward Handler**

Command:	Used to:
Routine Data	create a new routine (see page 8-10)
Instructions	view the instructions of the selected routine
Error Handler	view the error handler of the selected routine
Backward Handler	view the backward handler of the selected routine
In Module	view only the routines in the current module (see page 8-57)
In System	view all routines in all modules (see page 8-57)
Add/Remove Error Handler	add/remove an error handler to the selected routine (see page 8-53)
Add/Remove Backward Handler	add/remove a backward handler to the selected routine (see RAPID Reference Manual - *Programming off-line*)

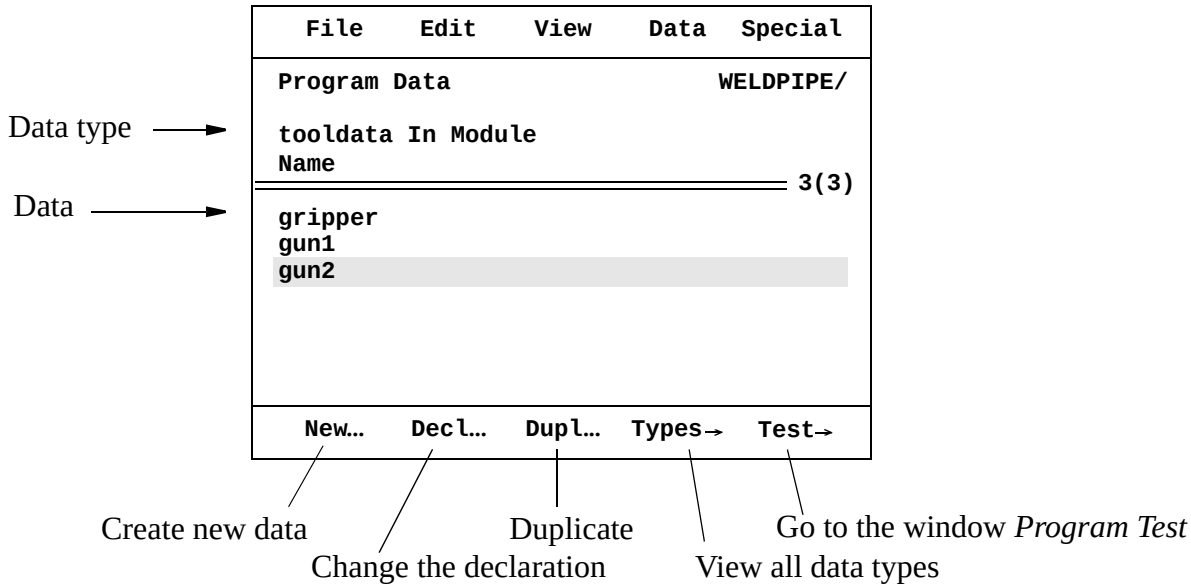
3.4.2 Menu: Special

Special

Mirror…

Command:	Used to:
Mirror	mirror a routine or a module (see page 8-39)

3.5 Window: Program Data



3.5.1 Menu: Data

Data

1 Value **2 Types**
3 In Module **4 In System** **5 In Routine “routine name”**
6 New Array

Command:	Used to:
Value	read or change the current value of selected data (see page 8-50)
Types	call up the list with all data types (see page 8-45)
In Module	call up only the data in the current module (see page 8-58)
In System	create new data (see page 8-47)
In Routine	call up all routine data
New Array	declare a new array data

3.5.2 Menu: Special

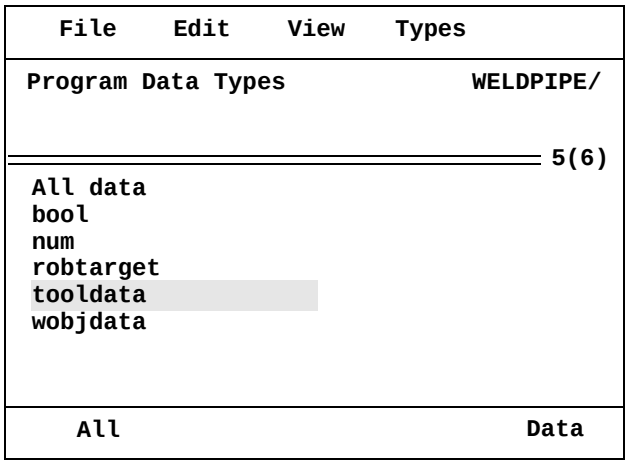
Special

1 Define Coord…
2 Go to selected position

Command:	Used to:
Define Coord	define a tool, work object or program displacement (see Chapter 10, Calibration)
Go to selected position	go to a selected position

3.6 Window: Program Data Types

Data types →



3.6.1 Menu: Types

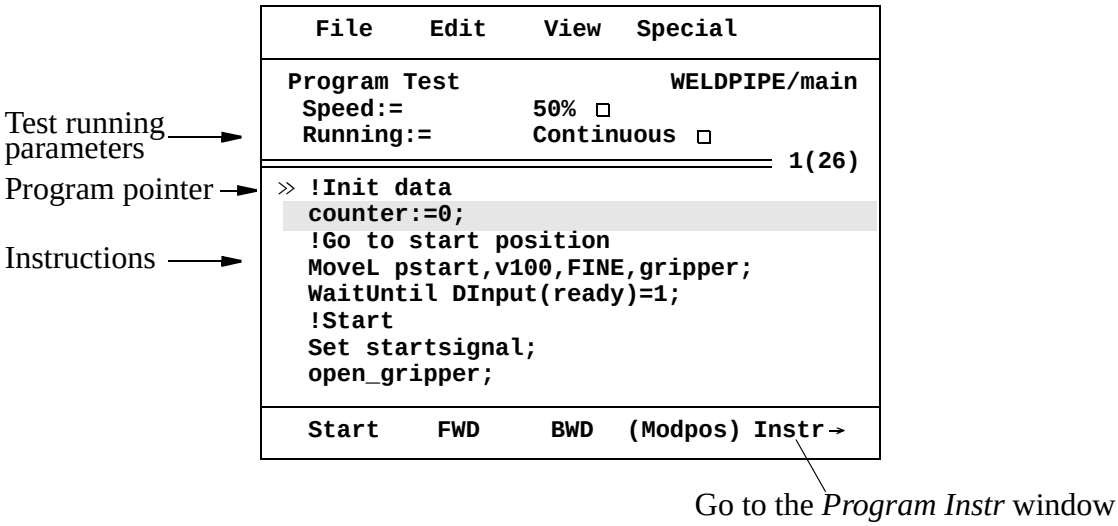
Types

1 Data

2 Used Types
3 All Types

Command:	Used to:
Data	call up all data of a selected type
Used Types	call up only those data types that are used
All Types	call up all data types (see page 8-45)

3.7 Window: Program Test



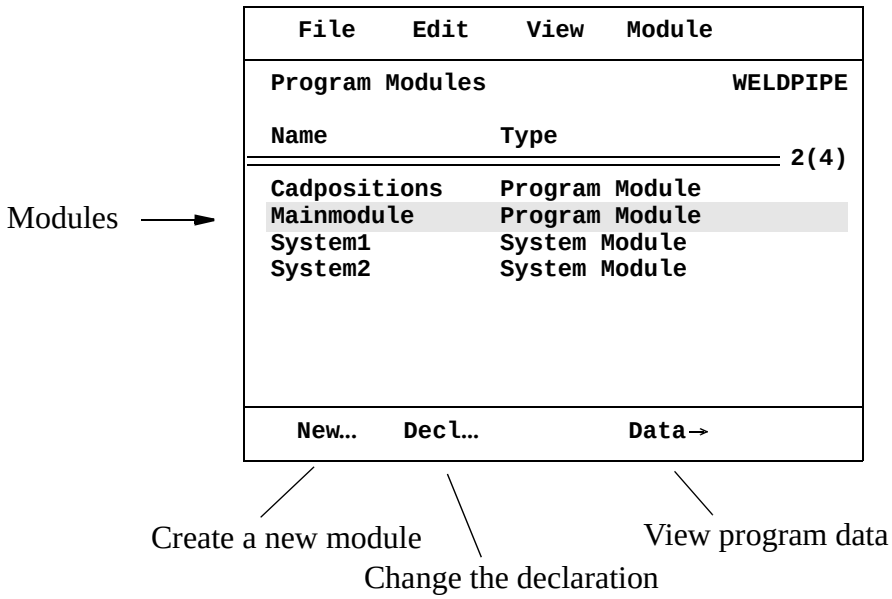
3.7.1 Menu: Special

Special

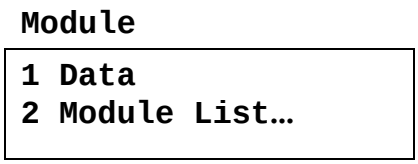
1 Move Cursor to PP **2 Move PP to Cursor** **3 Move PP to Main** **4 Move PP to Routine** **5 Call Routine…**
6 Call Service Routine…
7 Go to selected position…
8 Simulate…

Command:	Used to:
Move Cursor to PP	start from the latest stopped instruction (see page 8-25)
Move PP to Cursor	start from the selected instruction (see page 8-25)
Move PP to Main	start from the main routine (see page 8-25)
Move PP to Routine	start from any routine (see page 8-25)
Call Routine	start from any routine without loosing context (see page 8-26)
Call Service Routine	execute configured service routine without loosing context (see page 8-27)
Go to selected position	go to a selected position
Simulate	allow program execution in MOTORS OFF mode

3.8 Window: Program Modules



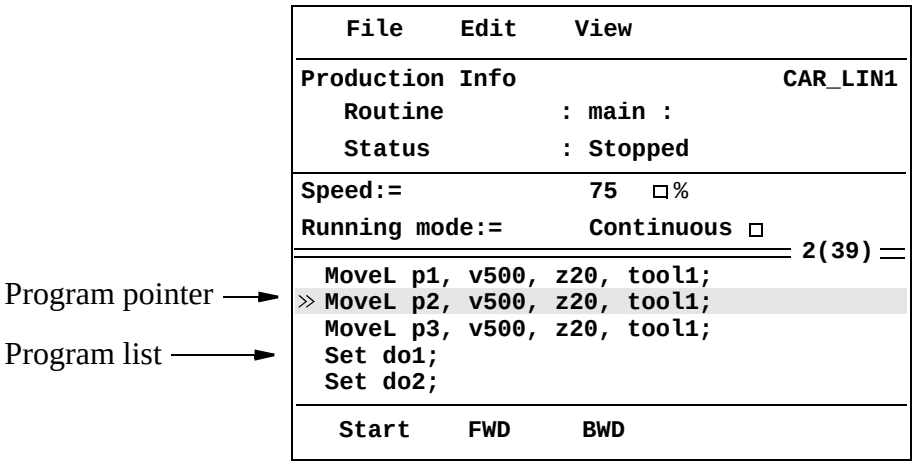
3.8.1 Menu: Module



Command:	Used to:
Data	view program data
Module List	view the complete module in a list (see page 8-59)

4 The Production Window

4.1 Window: Production



4.1.1 Menu: File

File

1 Load Program…

Command	Used to:
Load Program	load a program (see page 11-4)

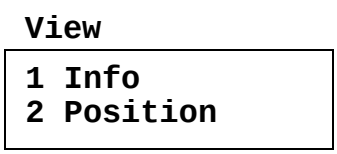
4.1.2 Menu: Edit

Edit

1 Goto…
2 Start from Beginning

Command	Used to:
Goto	go to a specific instruction
Start from Beginning	go to the first instruction in the program (see page 11-7)

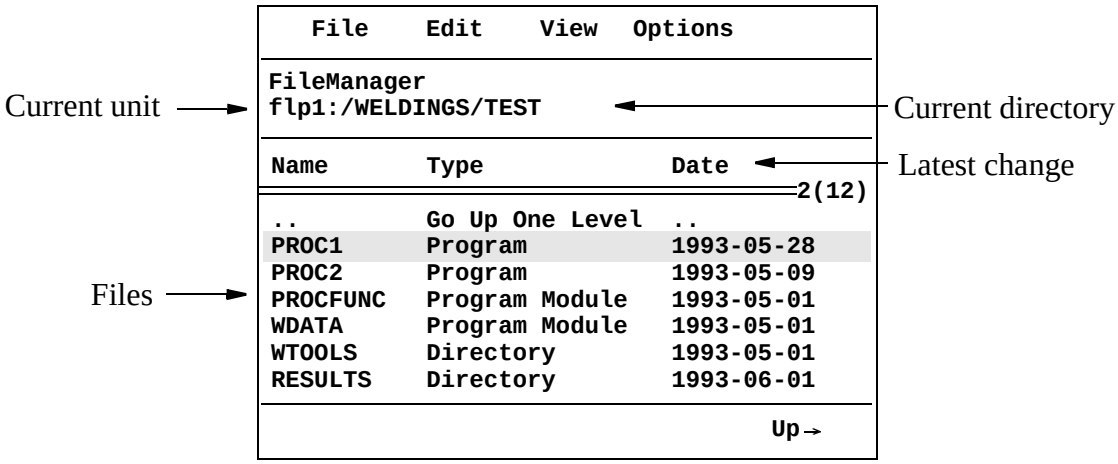
4.1.3 Menu: View



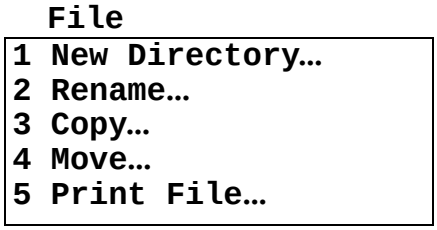
Command	Used to:
Info	display the program in the lower part of the window
Position	tune a position (see page 11-8)

5 The FileManager

5.1 Window: FileManager

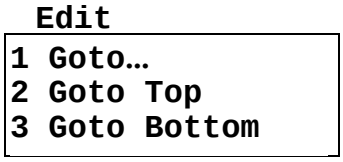


5.1.1 Menu: File



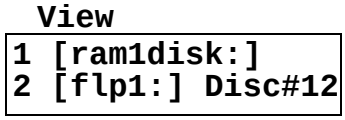
Command:	Used to:
New Directory	create a new directory (see page 13-5)
Rename	change the name of a selected file (see page 13-5)
Copy	copy a selected file or directory to another mass memory or directory (see page 13-6)
Move	move a selected file or directory to another mass memory or directory (see page 13-7)
Print File	print a file on a printer

5.1.2 Menu: Edit



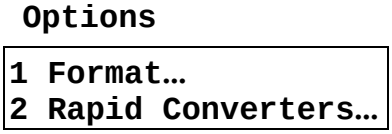
Command:	Used to:
Goto	go to a specific line in a list
Goto Top	go to the first file in a list
Goto Bottom	go to the last file in a list

5.1.3 Menu: View



Command:	Used to view:
ram1disk:	the files on the RAM disk (see page 13-4)
flp1:	the files on the diskette (see page 13-4)

5.1.4 Menu: Options

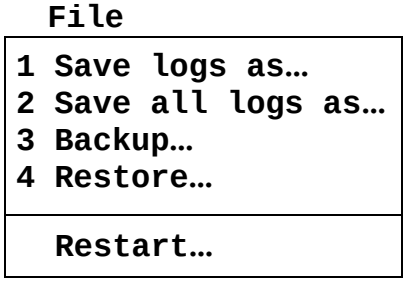


Command:	Used to:
Format	format a diskette (see page 13-7)
Rapid Converters	convert old program versions

6 The Service Window

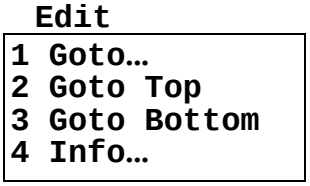
6.1 General menus

6.1.1 Menu: File



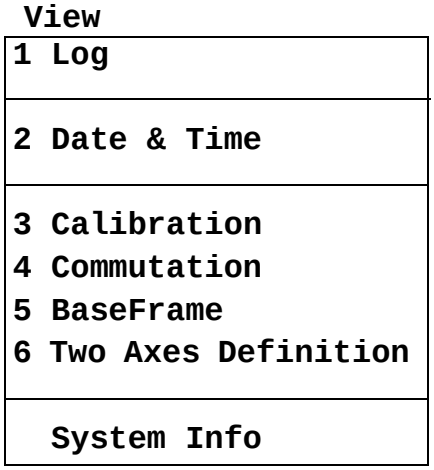
Command	Used to:
Save logs as	save logs on a diskette or other mass memory (see page 14-7)
Save all logs as	save all logs on a diskette or other mass memory (see page 14-7)
Backup	perform a backup (see page 14-10)
Restore	perform a restore (see page 14-11)
Restart	restart the robot (see page 14-11)

6.1.2 Menu: Edit



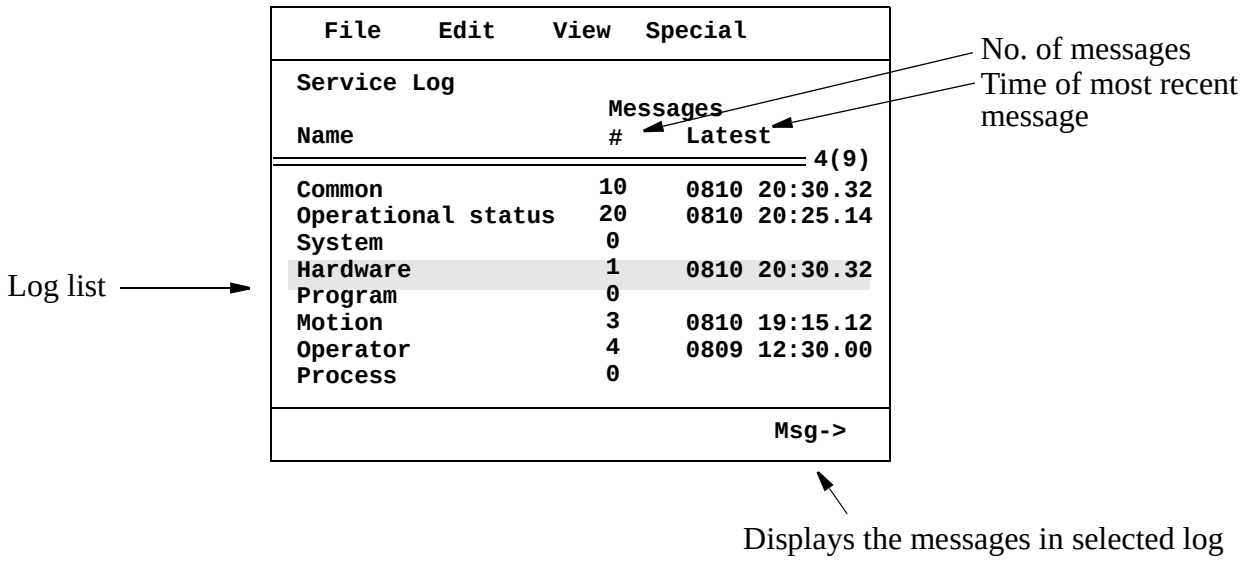
Command	Used to:
Goto	go to a specific line in a list
Goto Top	go to the first line in a list
Goto Bottom	go to the last line in a list
Info	view information about selected log messages (see page 14-6)

6.1.3 Menu: View

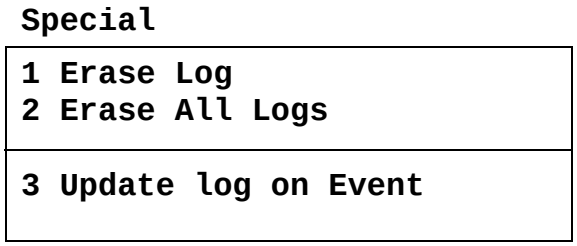


Command	Used to:
Log	display the different logs (see page 14-5)
Date & Time	set the date and time (see page 14-3)
Calibration	calibrate the robot (see page 14-8)
Commutation	commutate the motors (see The Product Manual/ *Repairs*)
BaseFrame	calibrate the base coordinate system (see Chapter 10, Calibration)
Two Axes Definition	calibrate the base coordinate system for a two axes manipulator (see Chapter 10, Calibration)
System Info	display system information (see page 14-9)

6.2 Window Service Log

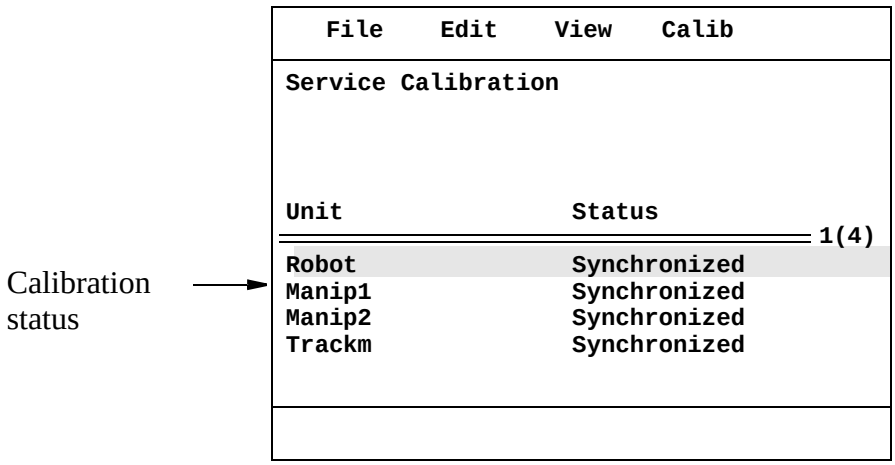


6.2.1 Menu: Special



Command:	Used to:
Erase Log	erase contents in selected log (see page 14-6)
Erase All Logs	erase contents in all logs (see page 14-6)
Update log on Event	update the log directly when a message is sent – the command is changed to **"Update log on Command"** when selected, which means that the log is not updated until the function key **Update** is pressed (see page 14-7)

6.3 Window Service Calibration



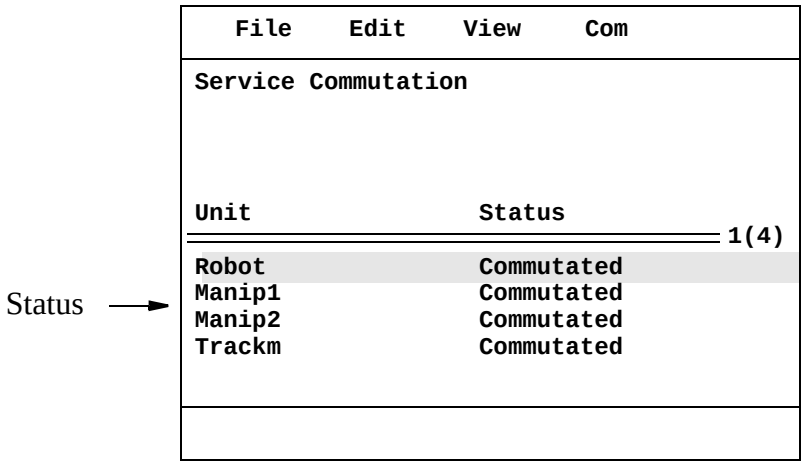
6.3.1 Menu: Calib

Calib

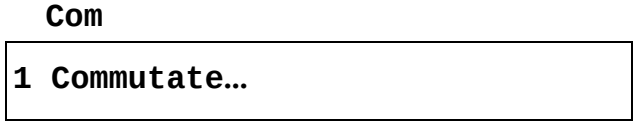
```
1 Rev.Counter Update…
2 Fine Calibrate…
```

Command:	Used to:
Rev.Counter Update	update the counter (see Chapter 10, Calibration)
Fine Calibrate	calibrate using the measurement system (see the Product Manual/*Repairs*)

6.4 Window Service Commutation



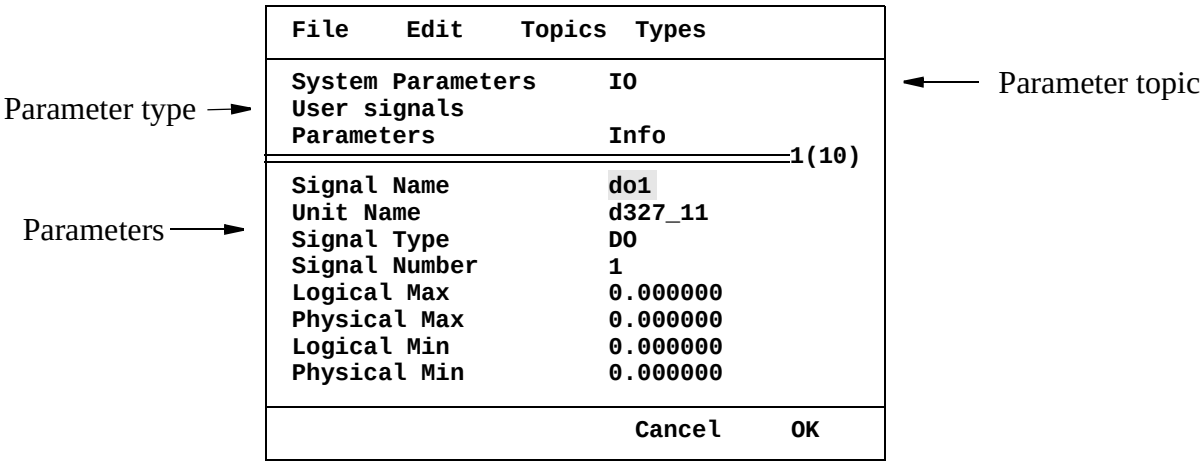
6.4.1 Menu: Com



Command:	Used to:
Commutate	commutate using the measurement system (see the Product Manual/*Repairs*)

7 The System Parameters

7.1 Window: System Parameters



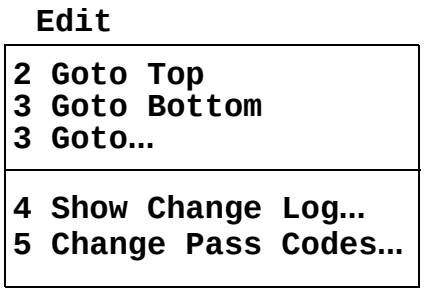
7.1.1 Menu: File

File

1 Load Saved Parameters…
2 Add New Parameters…
3 Save All As…
4 Save As…
5 Check Parameters
Restart…

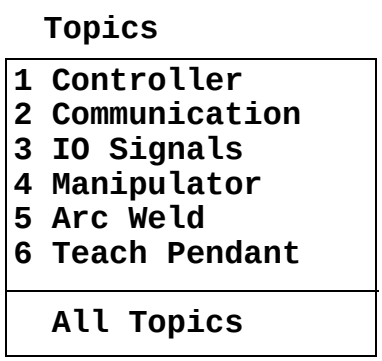
Command:	Used to:
Load Saved Parameters	load parameters from mass storage (see page 12-7)
Add New Parameters	add parameters from mass storage (see page 12-7)
Save All As	save all parameters on mass storage (see page 12-6)
Save As	save parameters on mass storage (see page 12-6)
Check Parameters	check parameters before restart (see page 12-5)
Restart	restart the robot (see page 12-4)

7.1.2 Menu: Edit



Command:	Used to:
Goto Top	go to the first line in a list
Goto Bottom	go to the last line in a list
Goto	go to a specific line in a list
Show Change Log	view information about the latest modifications made (see page 12-5)
Change Pass Codes	change pass codes (see page 12-46)

7.1.3 Menu: Topics



Command:	Used to view:
Controller	the parameter of the Controller topic (see page 12-37)
Communication	the parameter of the Communication topic (see page 12-31)
IO Signals	the parameters of the IO topic (see page 12-9)
Manipulator	the parameters of the Manipulator topic (see page 12-53)
Arc Weld	the parameters of the Arc Weld topic (see RAPID ProcessWare - *ArcWare*)
Teach Pendant	the parameters of the Teach Pendant topic (see page 12-45)
All Topics	all topics (see page 12-3)

7.1.4 Menu: Types

Content showing the types connected to the chosen topic.

INDEX

A

Add 9-14
add an instruction 8-16
Add New Parameters 12-7
Add or Replace Parameters 12-7
Align 6-5, 6-9
All Topics 12-3
All Types 8-46
analog output
  change manually 7-6
approach point 10-29
argument 8-13
  add optional 8-34
  change 8-33
arithmetic expression 8-18
Arm
  parameters 12-54, 12-55, 12-56
Arm check pnt
  parameters 12-58
Arm load
  parameters 12-57
array data 8-48
assignment 9-14
authorization 12-46
automatic mode 9, 5-4

B

base coordinate system
  define 12-61
  jogging 6-6
BaseFrame 10-9, 10-13, 10-18, 10-20
BWD 8-23

C

Calibration 10-6
calibration offset 12-53, 12-54, 12-78, 12-80, 12-81
calibration position
  define 12-54
Call Routine 8-26
Call Service Routine 8-27
calling a subroutine 9-10
change
  argument 8-33
  data 8-50
  displacement frame 10-43, 10-44
  instruction 8-29
  optional argument 8-34
  tool 10-29, 10-31
  work object 10-37, 10-38
Change Pass Codes 12-47
Check Program 8-21
choose
  routine 8-12
Clear 9-14
collision detection 6-5, 12-55
commutating 14-9
commutation offset 12-53, 12-54, 12-78, 12-80, 12-81
Compact IF 9-11
confirmation
  define 12-46
constant 8-45
Content 8-20
Controller
  parameters 12-37
coordinate system 4-22
coordinated motion 12-63
Copy
  File Manager 13-6
  instruction 8-21
copy
  data 8-50, 8-58
  files 13-6
  routine 8-11, 8-58
CPU Optimization 12-84
create
  data 8-47
  directory 13-5
  displacement frame 10-43
  module 8-56
  program 8-6
  routine 8-10
  tool 10-28
  work object 10-36
Cross Connections
  define 12-15
C-Start 14-11
Cut
  instruction 8-21

D

Data 8-45
data 8-45
    change 8-50
    create 8-47
    declaration 8-51
    delete 8-52
    duplicate 8-50, 8-58
Data Types 8-46
Date & Time 14-3
Declaration
    data 8-51
    module 8-56
    routine 8-35
Decr 9-14
define
    tool 10-28
Define Coord
    displacement frame 10-44
    tool 10-31
    work object 10-38
delete
    data 8-52
    file 13-6
    instruction 8-36
    module 8-57
    routine 8-36
digital output
    change manually 7-6
directory 13-3
    create 13-5
    delete 13-6
diskette 13-3
    format 13-7
displacement 9-6
displacement frame
    change 10-43, 10-44
Display 4-10
Duplicate
    data 8-50, 8-58
    routine 8-58
duplicate
    routine 8-11

E

elongator point 10-29
ELSE 9-13
Emergency stop 4-9
emergency stop 5-6
Enabling device 4-10
enabling device 5-5
Erase All Logs 14-6
Erase Log 14-6
Error Handler 8-52
error log 14-4
error management 18-21
error recovery 8-52
Event Routines 12-38, 12-39, 12-40, 12-41, 12-42
execution mode 8-23
expression 8-18
external axes
    defining 12-63
    jogging 6-16
external manipulator 12-62
external unit
    choose 6-16

F

Field 4-23
file 13-3
    copy 13-6
    delete 13-6
    move 13-7
    rename 13-5
File Extensions 12-45
file manager 13-3
file system 13-3
flp1 13-4
FOR
    change structure 8-35
format diskette 13-7
function 8-9
FWD 8-23

G

Go to selected position 8-27
group of I/O
    change manually 7-6
Groups
    parameters 12-14

H

Hide IPL 8-15

I

I/O
    parameters 12-9
I/O list
    define Most Common 7-4
IF 9-11
    change structure 8-35
In All Modules 8-58
In Module 8-58
Incr 9-14
incremental jogging 4-21
incremental movement 6-14
Info
    Service window 14-6
input signal
    define 12-12
inputs/outputs
    manual operation 7-3
Inputs/Outputs window 7-3
insert
    instruction 8-16
instruction 8-13
    add 8-16
    change 8-29
    copy 8-21
    delete 8-36
    move 8-21
instruction pick list 8-15
    Most Common 8-60
Instructions 8-12
IO Boards 12-9
IPL1 8-15
IPL2 8-15
I-Start 14-11

J

jogging 6-3
Joints 12-66
joystick 4-10, 6-4
Joystick lock 6-5

L

Load
    parameters 12-7
load
    module 8-57
    program 8-7, 11-4
Load Program 11-4
Load Saved Parameter 12-7
locking of joystick axes 6-5
Log 14-5
log 14-4
logical expression 8-18

M

Main Routine 8-12
main routine 8-5
Manipulator
    parameters 12-53
MANUAL FULL SPEED 4-9
manual mode 5-5
MANUAL REDUCED SPEED 4-9
Mark 8-30
Mirroring 8-39
modify
    argument 8-33
    data 8-50
    instruction 8-29
    position 8-31
ModPos 8-31
module 8-54
    create 8-56
    declaration 8-56
    delete 8-57
    open 8-57
    read 8-55
    save 8-58
Module List 8-59
Modules 8-55
Most Common
    I/O list 7-4
    instruction pick list 8-60
Motion Supervision 6-5, 12-55
Motor
    parameters 12-53, 12-54, 12-79, 12-81
Motors off 5-5
Motors on 4-9, 5-5
Move
    File Manager 13-7
move
    files 13-7
    instruction 8-21
Move cursor to PP 8-26
Move PP to cursor 8-26
Move PP to Main 8-26

Move PP to Routine 8-26
MoveC 9-3
MoveJ 9-3
MoveL 9-3

N

New 8-6
  module 8-56
New Directory 13-5
new routine 8-10

O

object coordinate system
  jogging 6-11
offset 9-6
Open
  module 8-57
  program 8-7
operating mode 5-4
operator dialogs 11-9
operator's panel 5-4
OptArg 8-34
optional argument 8-13
  add 8-34
Optional Package 12-45
output instruction 9-7
output signal
  define 12-12
override speed 11-5

P

parameters 12-3, 12-9, 12-62
pass code
  change 12-47
  define 12-47
Paste 8-21
persistent 8-45
position
  instruction 9-3
  modify 8-31
  read current 6-4
power failure 5-3
power supply 5-3
Preferences
  I/O window 7-4
  program window 8-60, 8-62
print
  I/O-list 7-8
  program 8-29
ProcCall 9-10
procedure 8-9
Production mode 4-9
Production window 11-3
program 8-5
  create 8-6
  load 8-7, 11-4
  print 8-29
  save 8-28
program data 8-5
Program Data Types window 8-46
Program Data window 8-45
program flow instructions 9-10
Program Instr window 8-12
program module 8-54
Program Modules window 8-55
Program Routines window 8-9
program running mode 11-5
Program Test window 8-22
program window 8-6
programming 8-5
Programming mode 4-9
P-Start 14-11

R

RAM disk 13-3
ram1disk 13-4
range of movement
  limit 12-54
read
  module 8-55, 8-57
  parameters 12-7
  program 11-4
Relays 12-68, 12-69
Rename
  file 13-5
reorienting the tool 6-9
Replace 8-37
required argument 8-13
Reset 9-7
reset
  emergency stop 5-6
restart 5-3, 12-4, 14-11
Rev.Counter Update 10-8

Robot
parameters 12-60, 12-61, 12-75, 12-76, 12-83, 12-84, 12-85, 12-86, 12-87
routine 8-5, 8-8
choose 8-12
create 8-10
declaration 8-35
delete 8-36
duplicate 8-11, 8-58
Routines 8-9
Running mode 12-51
running programs
production 11-3
testing programs 8-21

S

SafetyOpKey
parameter 12-37
SafetyRunChain
parameter 12-37
save
module 8-58
parameters 12-6
program 8-28
Save All As
parameters 12-6
Save As
module 8-59
parameters 12-6
Service window 14-7
Save Module 8-58
Save Module As 8-59
Save Program 8-28
Save Program As 8-28
ScreenViewer 15-15
ScreenViewer Window 15-15
Search 8-37
select several instructions 8-30
Selected Routine 8-12
service window 14-3
Set 9-7
SetAO 9-7
SetDO 9-7
SetGO 9-7
Show Change Log 12-5
Show IPL 8-15
signal
define 12-12
signal values
changing manually 7-6
Simulate wait 8-27
Single Type 12-66
speed correction 8-22
Start from Beginning 11-7
start program 11-6
starting program execution 8-24
start-up 5-3, 18-21
stationary tool
jogging 6-13
stopping program execution 8-25, 11-7
storage of program 13-3
store
module 8-58
program 8-28
string 8-18
subroutine 8-5, 8-8
call 9-10
System Inputs
define 12-22
system module 8-6
System Outputs
define 12-24
system parameters 12-3
change log 12-5
load 12-7
save 12-6

T

TCP 10-28
Teach Pendant 4-10
parameters 12-45
teach pendant 5-7
TEST
change structure 8-35
Test 8-22
Testing mode 4-9
text 5-9
time setting 14-3
tool
change 10-29, 10-31
define 10-28
tool coordinate system
jogging 6-7
tool reorientation 6-9

Transmission 12-57, 12-66
trap routine 8-9
trimming
    external axes 12-70, 12-71
typographic conventions 2-4

U

Unmark 8-30
Update log on Command 14-7
Update log on Event 14-7
User screen 15-15
user screen package 15-15
User Signals
    parameters 12-12

V

variable 8-45

W

wait
    a specific time 9-10
    for an input 9-8
WaitDI 9-8
WaitTime 9-10
WaitUntil 9-8
warning message 18-22
welcome window 4-15
work object
    change 10-37, 10-38
working space
    limit 12-54
world coordinate system
    define 12-61
    jogging 6-13

Glossary

Argument	The parts of an instruction that can be changed, i.e. everything except the name of the instruction.
Automatic mode	The applicable mode when the operating mode selector is set to .
Component	One part of a record.
Configuration	The position of the robot axes at a particular location.
Constant	Data that can only be changed manually.
Corner path	The path generated when passing a fly-by point.
Declaration	The part of a routine or data that defines its properties.
Dialog/Dialog box	Any dialog boxes appearing on the display of the teach pendant must always be terminated (usually by pressing ***OK*** or ***Cancel***) before they can be exited.
Error handler	A separate part of a routine where an error can be taken care of. Normal execution can then be restarted automatically.
Expression	A sequence of data and associated operands; e.g. *reg1+5* or *reg1>5*.
Fly-by point	A point which the robot only passes in the vicinity of – without stopping. The distance to that point depends on the size of the programmed zone.
Function	A routine that returns a value.
Group of signals	A number of digital signals that are grouped together and handled as one signal.
Interrupt	An event that temporarily interrupts program execution and executes a trap routine.
I/O	Electrical inputs and outputs.
Main routine	The routine that usually starts when the ***Start*** key is pressed.
Manual mode	The applicable mode when the operating mode switch is set to .
Mechanical unit	A group of external axes.
Module	A group of routines and data, i.e. a part of the program.
Motors On/Off	The state of the robot, i.e. whether or not the power supply to the motors is switched on.
Operator's panel	The panel located on the front of the control system.
Orientation	The direction of an end effector, for example.
Parameter	The input data of a routine, sent with the routine call. It corresponds to the argument of an instruction.
Persistent	A variable, the value of which is persistent.
Procedure	A routine which, when called, can independently form an instruction.

Program	The set of instructions and data which define the task of the robot. Programs do not, however, contain system modules.
Program data	Data that can be accessed in a complete module or in the complete program.
Program module	A module included in the robot's program and which is transferred when copying the program to a diskette, for example.
Record	A compound data type.
Routine	A subprogram.
Routine data	Local data that can only be used in a routine.
Start point	The instruction that will be executed first when starting program execution.
Stop point	A point at which the robot stops before it continues on to the next point.
System module	A module that is always present in the program memory. When a new program is read, the system modules remain in the program memory.
System parameters	The settings which define the robot equipment and properties; configuration data in other words.
Tool Centre Point (TCP)	The point, generally at the tip of a tool, that moves along the programmed path at the programmed velocity.
Trap routine	The routine that defines what is to be done when a specific interrupt occurs.
Variable	Data that can be changed from within a program, but which loses its value (returns to its initial value) when a program is started from the beginning.
Window	The robot is programmed and operated by means of a number of different windows, such as the Program window and the Service window. A window can always be exited by choosing another window.
Zone	The spherical space that surrounds a fly-by point. As soon as the robot enters this zone, it starts to move to the next position.